Switching algorithm for an all-optical packet switch using fully shared buffer architecture with and without priority traffic: A simulation study

by
Hazari Praveen Kumar

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

February, 2000

Switching algorithm for an all-optical packet switch using fully shared buffer architecture with and without priority traffic: A simulation study

A Thesis Submitted
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF TECHNOLOGY
By
Hazari Praveen Kumar

to the
DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
February, 2000

Certificate

This is to certify that the work contained in the thesis entitled ***Switching algorithm for an all-optical packet switch using fully shared buffer architecture with and without priority traffic: A simulation study,*** by Mr. Hazari Praveen Kumar (Roll no. 9810423) has been done under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Yatindra Nath Singh,
Assistant Professor,
Dept. of Electrical Engineering,
Indian Institute of Technology,
KANPUR – 208 016.

Dedicated

to

My parents

Acknowledgement

I take this opportunity to express my sincere gratitude to my thesis supervisor Dr. Y.N.Singh for his excellent guidance during my Thesis work. He was always available to help, guide, and encourage me. His belief in my endeavor gave me enormous confidence to reach my goal.

I am thankful to my batchmates Rao, Murthy, Gaurav, Dhaval, Harish, Nitin, Bhuvenendra and Alpna for maintaining a wonderful work atmosphere at ernet lab. I would like to mention the help and support of Vineet, Amitabh, Vivek, Manoj, and all the research staff of network floor. Thanks to M.Murali, Ravi, T.Murali, Santosh Kulkarni and vadayar for making my stay at Hall 5 memorable. Thanks to IIT Kanpur for providing excellent facilities for research.

Words are not enough to explain my feeling towards my Parents, Sisters. I always found them standing by my side during my hard days. They have always been the constant source of inspiration for me.

-praveen

ABSTRACT

In first generation optical networks which are in existence today, the electronics at a switching node not only handle all the data intended for that node i.e., header part, but also all the data that is passing through that node i.e., payload part. Researchers realized that if the later part i.e., payload part could be routed through in optical domain, the burden on the underlying electronics at the node would be reduced and thereby one can achieve switching at higher speeds. This has led to 'all-optical' switches, in which data portion of a packet remains in optical format from the source to destination, but header part will be processed using optical or optoelectronic techniques. Presently there exists a limit on the number of wavelengths that can be multiplexed using WDM (Wavelength Division Multiplexing) and hence on the number of buffers that can be used in optical domain. So there is a need for having switching strategies which use~~s~~ buffer efficiently. Keeping this facts in mind switching strategies have been proposed in this work for an all-optical packet switch using shared buffer architecture with and without priority traffic.

Contents

List of figures

List of abbreviations

ATMOS	- Asynchronous Transfer Mode Optical Switching
ASE	- Amplified Spontaneous Emission
BISDN	- Broadband Integrated Services Digital Network
EDFA	- Erbium Doped Fiber Amplifier
FIFO	- First In First Out
HOL	- Head Of Line
LAN	- Local Area Network
LIFO	- Last In First Out
QoS	- Quality of Service
RACE	- Research and development in Advanced Communications in Europe
RAM	- Random Access Memory
SOA	- Semiconductor Optical Amplifier
TOAD	- Terahertz Optical Asymmetric Demultiplexer
WAN	- Wide Area Network
WDM	- Wavelength Division Multiplexing

Chapter 1
Introduction

1.1 Photonic switching

The existing telecommunication networks, whether circuit switched or packet switched, are oriented towards particular applications. Thus, we have different networks for voice, video and data applications operating in parallel and independently. While each of these networks is suitable for the application it is designed for, they are not very efficient for supporting other applications. Requirement of broadband integrated communication system, which can accommodate a variety of diverse services with different bandwidth, has been recognized for quite a long time. The range of future services will be very diverse in terms of channel occupancy (continuous or bursty), channel capacity required, connection setup time, connection duration and bandwidth used [4]. Broadband Integrated Services Digital Networks (BISDN) are expected to provide these services. Few examples of such services are high-speed computer file transfers, interactive computer data exchange, and multimedia connections (e.g. video on demand) [2]. These networks and nodes are capable of sharing the bandwidth between different type of users. Main requirements of BISDN are wide bandwidth, high speed and fast switching, which are difficult to meet with electronic technology. Photonic technology has the capability to fulfil these requirements and presently numerous efforts are being made to exploit it [1].

Optical fiber has replaced cables as transmission media for the most of the long distance Wide Area Networks (WANs) as well as Local Area Networks (LANs) as they provide large bandwidth with virtually no loss. Optical to electrical and electrical to optical conversion limits bit rates that can be achieved in fiber link. It also limits the bit rates with which switching node operates. Optical fiber can provide capacity of some thousands of GHz [5]. This capacity of optical fiber is not used fully due to switching speed at the switching nodes in the present electronic switches. This has led to "all-optical" (photonic) switches. The term "all-optical" implies that the data portion of a packet remains in optical format from the source to the destination. While the data remain all optical, both optical and optoelectronic techniques have been used to process packet routing functions. Use of optical switching will make it possible to have ultra high bit rates in optical fiber, thereby utilizing the available capacity.

1.2 Photonic packet switching

Packetized communication is the transportation of data as a discrete unit or packet accompanied by some form of routing information. Packetized communication has received importance due to the reason that it uses resources efficiently compared to circuit switching, and also because it will carry data in some specified sizes of packets irrelevant of what data it is carrying, thereby providing uniformity which is required in broad band data transfer applications.

Photonic packet switching has recently received increased attention due to rapid growth in Internet traffic and the need for next generation Internet technologies [8]. The proliferation of data and packet switched networks in our everyday activities and commerce has raised the importance of advancing research beyond circuit switched optical networks to support packet switching. Photonic packet switches offer high speed (i.e., data rate), format transparency, and flexibility required by future communication networks. Packet switching requires delay elements (i.e., buffering) at the switching node to save packets contending for the same output line. Generally a fiber having length equal to the necessary delay is used to provide delay in the present photonic switching nodes. In these delay elements, once the signal enters the fiber it will come out of it after a fixed interval of time [6]. Purely high-speed photonic packet switching is today limited by the difficulty of synchronizing, decoding and buffering directly in the photonic domain. Presently many architectures are under investigation for all-optical (photonic) packet switching. Key technology issues involved in photonic packet switching [8] are:

- packet generation
- header coding
- optical buffering
- synchronization
- regeneration

- fast optical switching

In this work we have concentrated on switching strategies with and without priority using photonic packet switching architecture based on multiwavelength fiber loop memory proposed in RACE (Research and development in Advanced Communications in Europe) ATMOS (ATM Optical Switching) project [2].

1.3 A comparison of photonic and electronic packet switches

There are some important differences between photonic and electronic switch technologies [10]. One distinction between switching signals electronically and optically is that electronic data can be stored statically, while optical data can not be stored statically and must be processed and switched on the fly. Other factors that influence a packet switch architecture come from the relative cost of switch facilities versus bandwidth. Electronic packet switching offers advantages in terms of cost, as switch crosspoints and memory needed in the switch are easily available in the integrated forms as RAMs (Random Access Memory). Therefore, routing techniques were developed that heavily utilized memory for synchronization, flow control, and contention resolution. But the switching speeds that are possible with these switches are limited and hence with these switches link bandwidths are not used efficiently. The situation is reversed for photonic switching systems. The link bandwidth of optical fibers is relatively inexpensive, whereas photonic switch elements, optical memories are more expensive than their electronic counterparts.

The buffer size or depth of an electronic switch is chosen to handle the mismatch between the input flow of packets. Buffering is also used to pipeline packets through the switch, providing an increase in throughput with a trade off in latency. From broad band communications perspective electronic buffers can be a limiting factor. Photonic switch research has been presently focused on how to best utilize the wide bandwidth to increase performance or fulfill the requirements, which are difficult to be achieved with electronic switching. One example is the capability of photonic switches in conjunction with optical fibers to maintain data in all-optical format from the source to the final destination. This characteristic allows the simultaneous transport of multiple data rates (data-rate transparency) and multiple formats (format transparency).

Photonic packet switches are characterized as elastic-buffered or passive-buffered. Here, the term passive-buffered implies that only passive delay lines are used, typically fiber loops. The primary type of optical buffer in use recirculates packets instead of holding them statically or stationary in memory. It is also possible to store multiple packets in a single fiber loop using WDM technology. Due to noise accumulation in the fiber loop memory the depth of fiber loop memory can not exceed certain number of recirculations. This implies that we can not save more than some number of packets for any particular output. The noise accumulation in the fiber loop is proportional to the number of wavelengths used in the fiber loop, which are used to save the packets in the case of contention. This causes a limitation on the number of wavelengths that can be used in the fiber loop. Hence there is a need of algorithms for photonic switching that uses buffer efficiently.

1.4 Necessities of priorities in a packet switched network

As mentioned in section 1.1 future broadband networks are required to provide a wide variety of services with very different bandwidth and quality of service (QoS) requirements. Because of statistical multiplexing of a large number of sources in these applications, congestion may arise in these networks due to overlap of peak rate of different sources in time. This could lead to the possibility of cells being lost inside the network or being delayed making them useless at receiver. This will happen specifically in real time services. The main QoS requirements deal with cell loss rate and transfer delay characteristics.

One way to provide different QoS is by carefully dimensioning the network to satisfy the most stringent QoS requirement. Such an approach leads to poor utilization of network resources. A more flexible approach is to provide some priority mechanism inside the network. Priorities can be of two types: delay (or time) priorities and loss (or space) priorities. Delay priorities provide preferential services to some classes of traffic in order to control their end-to-end delay and delay variations. Loss (or space) priorities provide preferential access to buffer space. In this work, we shall deal only with loss priorities.

1.5 Objective of the thesis

In this work switching algorithm for an all-optical packet switch based on multi-wave length fiber loop memory is considered so that we can use buffers (wave lengths in the fiber loop) efficiently. We have implemented two strategies for switching within the photonic switch based on multiwavelength fiber loop memory.

1. In the first strategy we have considered the case in which packets are dropped in the fiber loop memory due to accumulation of noise, after rotating 'recirculation limit' number of times. We call this strategy as 'dropping packets inside', as the packets are lost inside fiber loop after rotating 'recirculation limit' number of times.

2. In the second strategy controller will drop newly arrived packets at the inputs so that in the buffer at the end of any time slot there are no packets for a particular output exceeding 'recirculation limit'. We will call this strategy as 'dropping packets at the input', because packets for a particular output are limited at the inputs of the switch. This strategy is supposed to use the buffer more efficiently. From simulation of this two strategies parameters of interest for a packet switch such as packet loss probability, delay experienced by a packet in the switch are calculated and compared.

We have also proposed priority switching algorithm for the all-optical packet switch based on multiwavelength fiber loop memory, for two classes of arrivals one which is loss sensitive and having high priority, another is loss insensitive and having low priority [13,15]. In this algorithm we will try to save high priority packets as far as possible, From simulation of this algorithm packet loss probability, average delay for both the classes of arrivals are calculated. And it is shown that high priority packets suffer low packet loss probability compared to low priority packets. This strategy is similar to 'push out algorithm', as low priority packets are pushed out from buffer once newly arrived high priority packets finds place in shared buffer and total number of packets for a particular output exceeds 'recirculation limit'. The scheme is not exactly same as 'push out algorithm' discussed in literature [13] due to specific characteristics of optical switch. High priority packets on arrival will be lost if they do not find free buffers. While in 'push out algorithm' low priority packets are removed and in their place high priority packets are filled in the same time slot. This is not possible with photonic switch based on multiwavelength fiber loop memory. Detailed discussion about this is given later.

1.6 Organization of the thesis

In Chapter 2, functions of a photonic switch are discussed. In Chapter 3, architecture of the photonic switch based on multiwavelength fiber loop memory, and little description of components used in the switch is given. In Chapter 4, model of the switch used for the simulation purposes is given. Then two switching strategies for the switch are discussed, (a) 'Drop inside' strategy and (b) 'Drop at inputs' strategy. In Chapter 5, priority switching algorithm for the switch is given. Chapter 6 presents results and performance of the switching strategies. Finally Chapter 7 focuses on the conclusion and future scope of this work.

Chapter 2
Functions of a photonic packet switch

Important functions that need to be performed at a photonic packet switching node [2, 10, 11] are

- Synchronization of incoming packets
- Header recognition
- Routing
- Flow control and contention resolution

Generally first function is performed by an unit known as link processor unit, Thus link processor unit synchronizes the incoming packets. Switching fabric performs remaining functions. Switching fabric processes headers of the arriving synchronous packets at its input and routes them depending on the header information. Switching fabric uses buffers to store the packets contending for same output, so that they will be transmitted in successive slots. It will drop packets if sufficient buffers are not there, t ~~T~~hereby achieving flow control.

2.1 Synchronization

Synchronization is the process of aligning two pulse streams in time. In photonic packet-switching networks (we are assuming packets of fixed size), it can refer either to the alignment of an incoming pulse stream and a locally available clock pulse stream or relative alignment of two incoming pulse streams. The function of synchronizer can be understood from figure 2.1. The two periodic pulse streams, with period T, shown in figure 2.1(a) are not synchronized because the top stream is ahead in time by ΔT. In figure 2.1(b), the two pulse streams are synchronized.

Thus to achieve synchronization, the top stream must be delayed by ΔT with respect to the bottom stream. This delay is not fixed one. Synchronizer in the photonic switching node requires a tunable delay element, since the amount of delay that has to be introduced is not known a priori. Thus we will now discuss how tunable optical delays can be realized.

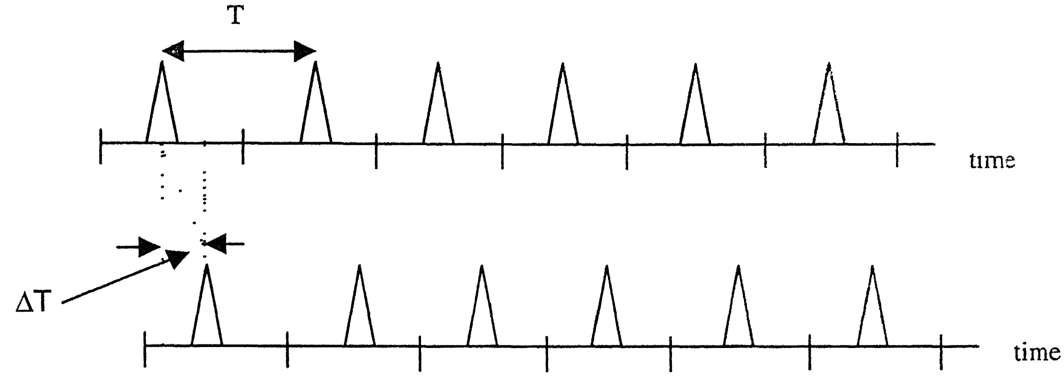


Fig 2.1 (a) The two periodic pulse streams with period T are out of synchronization; the top stream is ahead by ΔT.

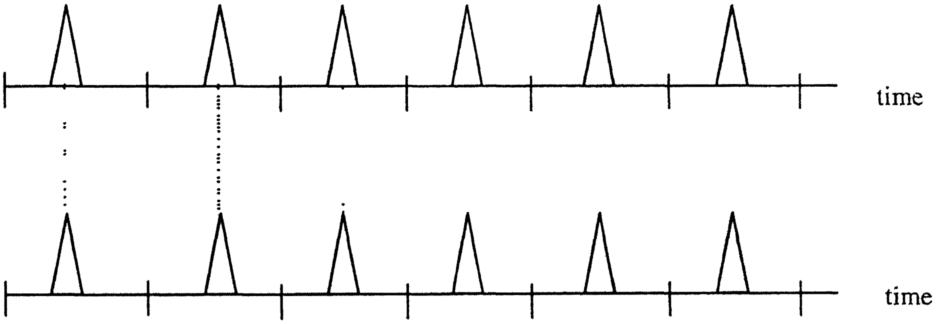


Fig 2.1(b) The two periodic streams have been synchronized by introducing a delay ΔT in the top stream relative to the bottom stream.

Tunable delays

A tunable optical delay line capable of realizing any delay, in excess of a reference delay, from 0 to $T(1- 2^{-k})$, in steps of $T/2^k$, is shown in figure 2.2. The parameter k controls the resolution of the delay achievable. The delay line consists of k fixed delays with values $T/2, T/4,\ldots,T/2^{k-1}$, $T/2^k$ interconnected by (k+1) 2×2 optical switches, as shown. By appropriately setting the switches in the cross or bar state, an input pulse stream can be made to encounter or avoid each of these fixed delays. If all the fixed delays are encountered, the total delay suffered by the input pulse stream is $T/2+T/4+\ldots+T/2^{k-1}+$

$T/2^k = T(1-2^{-k})$. This structure can be viewed as consisting of k-1 stages followed by an output switch as shown in figure 2.2.

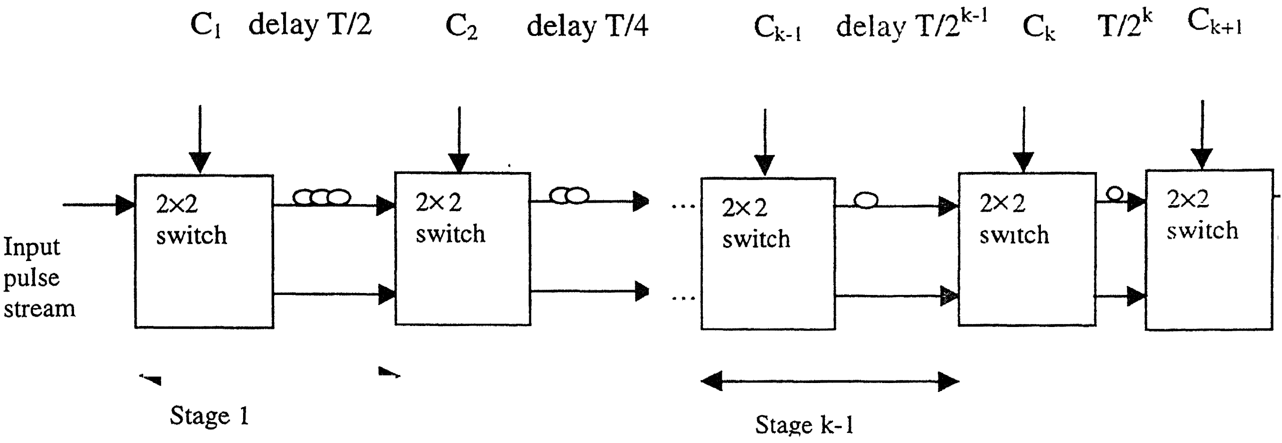


Fig 2.2 A tunable delay line capable of realizing any delay from 0 to $T(1-2^{-k})$, in steps of $T/2^k$.

The output switch is used to ensure that the output pulse stream always exits from the same port. C_1, C2,......, C_{k-1}, C_k, C_{k+1} are control inputs. With a tunable delay line like the one shown in figure 2.2, two pulse streams can be synchronized to within a time interval of $T/2^k$. The value of k, and thus the number of fixed delays and optical switches, must be chosen such that $2^{-k}T << \tau$, the pulse width. Given a tunable delay, the synchronization problem reduces to one determining the relative delay, or phase, between two pulse streams. A straightforward approach to this problem is to compare all shifted versions of one stream with respect to the other. The comparison can be performed by means of a logical AND operation. This is somewhat expensive approach. An alternative approach is to use an optical phase lock loop to sense the relative delay between the two-pulse streams [11].

Generally framing pulses are used to mark the packet boundaries (we are assuming fixed size of packets), the framing pulses must occur periodically. Delaying one stream with respect to another as discussed above can carry out synchronization of packets.

2.2 Header recognition

Control unit in the switching fabric performs this function. For a header of fixed size, the time taken for detecting and processing the header is fixed, and the remainder of the packet is buffered optically using a delay line of appropriate length. The processing of the header bits may be done electronically or optically, depending on what kind of processing is needed and the kind of control input required by the switch. Electrically controlled switches employing the electro-optic effect and fabricated in lithium niobate are most commonly used in switch based network experiments today. In this case, the header processing can be carried out electronically after the header bits have been demultiplexed into a parallel stream. The packet destination information from the header is used to determine using a look-up table the outgoing link of the switch for this packet. Conflict occurs if more than one inputs have a packets destined for the same output. This is the reason for having buffers in the routing node. The header information can be read by demultiplexing the header by using TOAD (Terahertz Optical Asymmetric Demultiplexer) [11]. However, this is a relatively expensive way of reading the header, which is a task that is easier done with electronics than optics. With this in view, several techniques have been proposed to simplify the task of header recognition. One common technique is to transmit the header at a much lower bit rate than the packet itself, allowing the header to be received and processed relatively easily within the routing node. The packet header could also be transmitted on a wavelength that is different from packet data [11]. It could also be transmitted on a separate subcarrier channel on the same wavelength. All these methods allow the header to be carried at a lower bit rate than the high–speed data in the packet; allowing for easier header recognition. Some times there is a need to replace existing header with another header as done in ATM switches. This function is known as header regeneration.

Header regeneration

Header regeneration is the process of computing, generating, and reinserting a header with the associated payload at the appropriate switch output port [10]. There are several circumstances where this functionality is required. A) in all-optical photonic switches where the header is completely removed from the payload for processing (e.g., multiwavelength out-of-band-signaling) or B) where routing strategies require a modification of the packet header (e.g., cell routing in the ATM switches). Many header replacement techniques for all optical packet switches have been proposed in literature [16].

2.3 Routing

Routing is the method used to choose the preferred path to send a packet from its source to destination, whether it be input and output ports for a switch or end points in a multi switch network. Routing control may be centralized or distributed. Centralized control involves a single processor monitoring the network and setting up the switching states according to the routing requests. As networks become more widely distributed or incorporate more switches, centralized control degrades latency and throughput, and increases processing complexity. With distributed routing, packets carry destination information for processing at each node. This form of processing reduces the burden on a centralized processor and increases the switch throughput. Additionally, distributed routing decisions are local information, whereas with centralized routing, decisions in a wide area network are made on global and perhaps obsolete information. Mixtures of centralized and distributed routing may provide optimal performance since global information is often useful in computing routing path [10].

A routing protocol must be able to handle both switch level routing and contention resolution. Centralized and distributed switches can be classified as randomly (or semi randomly) connected or regularly connected. In real world applications, distributed switched networks evolve in a semi-random manner, where a mathematical relationship between the addresses of neighboring nodes are not guaranteed, and packet destination address must be .carried throughout the routing process. In this case, the destination address is mapped at each switch to a local switch output port. Generally, routing tables are maintained for this mapping. Electronically implemented routing tables grow in complexity and access delay with increasing network size can pose a switch bottleneck; therefore, parallel access optical and optoelectronic routing tables are desirable to maintain high switch performance.

2.4 Flow control and contention resolution

Traffic in the switch and network must be regulated to prevent packets from running into each other or congesting resources. Contention resolution is needed to mediate flow of packets through internal switch links and switch output ports. In general, a routing node contains buffers to store the packets from the incoming links contending for the same output links. These packets are forwarded to output links in successive time slots and thereby contention is resolved. Buffers in the case of electronic switches are realized using Random Access Memory (RAM). This is unlikely to be practical, or even feasible, in photonic domain [11]. Presently optical buffers are generally realized from a length of fiber equal to the necessary delay. In these loops packets recirculate instead of staying stationary. There are buffering strategies that are used in general for packet switches and one need to investigate their applicability in photonic domain [6,9,12].

A) Input buffering

With input buffering switch (figure 2.3) an arriving cell enters a first-in first-out (FIFO) buffer located at its port of entry, if space is available. In each time slots, the switch resolves contentions priory to switching. If all head-of-line (HOL) cells are destined to distinct output ports, then all of them are admitted and switched to their desired output lines. However, if K HOL cells ($1< K \leq N$) are destined to a particular output port, only one cell is chosen, according to some selection policy, to be switched and other cells wait to participate in the next time slot selection process. Several selection policies have been proposed in the literature including the following [9,12]:

- Random selection
- HOL FIFO (or oldest HOL) selection
- HOL LIFO (last-in first-out) selection
- Global FIFO (or earliest arrival) selection
- Longest queue selection
- Oldest queue selection
- Cyclic selection
- Prediction based selection

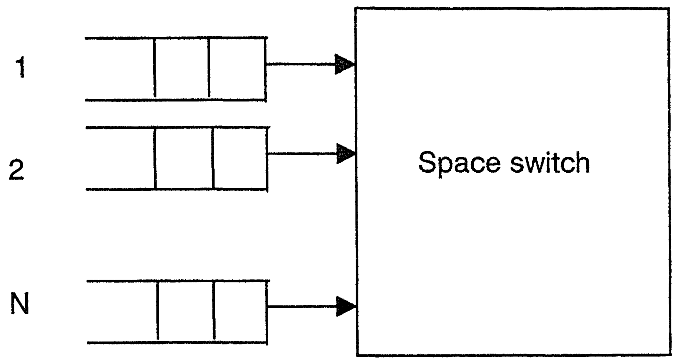


Fig 2.3 An input buffered switch.

Input buffering switches with FIFO queueing suffer from HOL (Head Of Line) blocking. In any given time slot while a cell is waiting for its turn to access an output port, other cells may be blocked behind it despite the fact that their destination ports are possibly idle. The maximum throughput of these switches is limited to 0.586 [6, 9, 17] (for uniform traffic) regardless of specific selection policy. Input buffering is never proposed for purely optical implementation, primarily because of its poor performance. It is not possible to implement different selection policies, as they will require optical processors, which are difficult to implement [6].

B) Output buffering

`In a timeslot there may be two or more packets contending for the same output port. In such a situation, given the assumption that input and output lines operate at the same speed, each output line can serve only one cell per time slot, hence other cells must be buffered. Output buffering as shown in figure 2.4 is buffering of packets at the output of the switch. Output buffering requires switch to be operated at a speed equal to N times the speed with which packets are coming on inputs. However, only one packet will be served by an output line (assuming inputs and output are operating at same speed) in each time slot and other cells with same output requests have to be buffered, if space is available. There is no problem such as HOL blocking for output buffering. Output buffering is the basis of many optical packet switches, although it is usual for an optical packet switch to emulate an output buffered switch rather than have separate identifiable output buffers. Once a packet is in a fiber loop delay line, it is not possible to change its

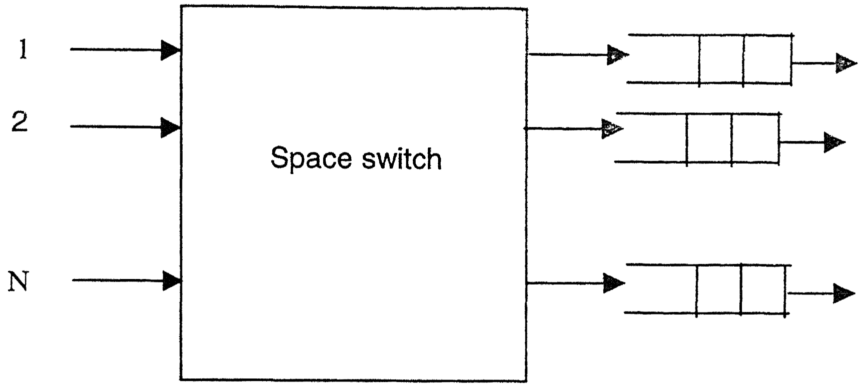


Fig 2.4 An output buffered swıtch

delay by removing it before it reaches the end. This suits output bufferıng where the delay in each output buffer can be determined before the packet enters ıt [6]

C) Shared Buffering

Electronic shared buffering may be regarded as a form output buffering, where all theoutput buffers share the same RAM (memory) area. Hence, the capacity restriction is not on the number of packets in each individual buffer but on the total number of packets in all buffers. Shared buffering is one of the most common methods of implementing electronic ATM switches. Usually electronic ATM switches are implemented as shared electronic RAMs as shown in the figure 2.5. Here we have shown a shared buffer switch, which is having 'B' buffer positions, each can save a packet. Packets are assumed to be

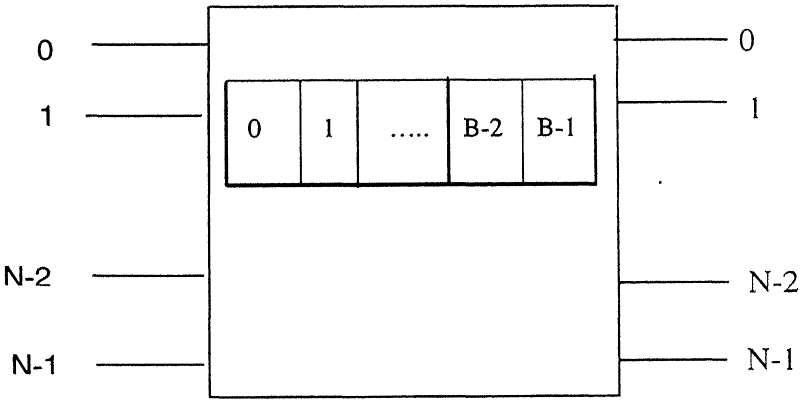


Fıg 2.5 A shared buffer switch

of fixed size. When this scheme is to be implemented in optical domain, certain problems are present. One of the problems is non-availability of optical RAM, which can replace electronic RAMs. However, many optical packet switches use shared buffering when emulating output buffering, since the delay lines are shared among multiple output lines. The photonic switch under study which is based on multiwavelength fiber loop is modeled as shared memory switch with the constraints that

1. A buffer position can not be erased and written in to simultaneously in the same time slot. If electronic RAMs are used, this restriction is not there.
2. There exists a limit on the number of recirculations after which it is not possible to recover original signal at the switch output due to noise accumulation within the loop.

This will be discussed later in chapters 3 and 4.

D) Recirculation buffering

In recirculation buffering (shown in figure 2.6), a number of recirculation loops from the output of space switch are fed back to the input. Each loop has a delay of one packet. If more than one packet arrives at the input of space switch for a particular output, all but one are placed in to the recirculation loops.

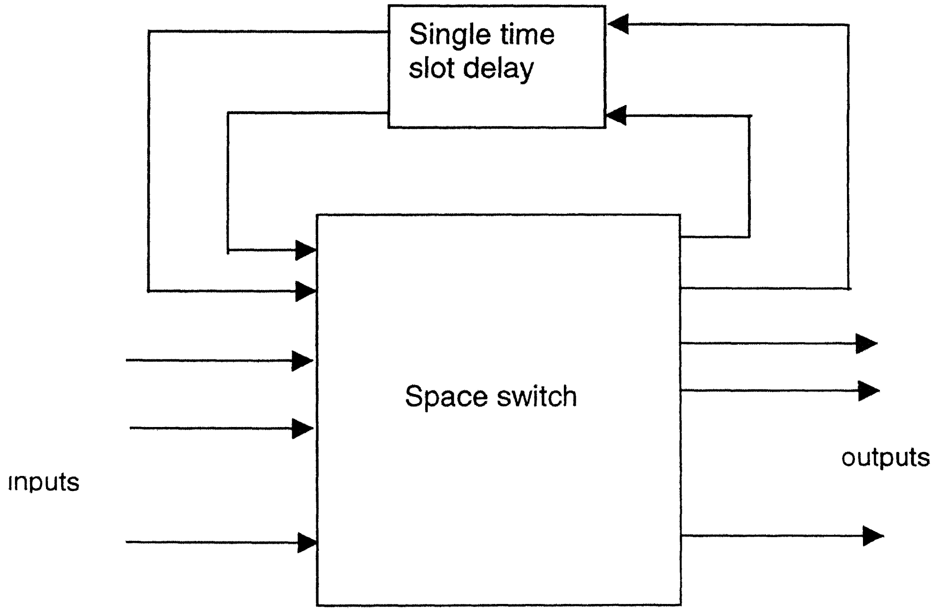


Fig 2.6 switch employing recirculation buffer

When recirculation loops are implemented optically with unity delays, many recirculations are required, implying high loss and accumulation of amplifier noise in the loops. Since optical amplifiers are needed to compensate the attenuation and other losses in loop [6].

Chapter 3

Photonic switch based on multiwavelength fiber loop memory

3.1 Switch architecture

This switch architecture has been proposed in ATMOS (Asynchronous Transfer Mode Optical Switching) RACE (Research and development in Advanced Communicatıons in Europe) project [2]. Photonic switch based on multiwavelength fiber loop memory switching fabric consists of two main functional blocks as shown in fıg 3.1.

1. A set of link interfaces, which perform signal regeneration and clock synchronization at the switching node.

2. Switching fabric, which is responsible for the routing, header recognition, flow control and contention resolution functions.

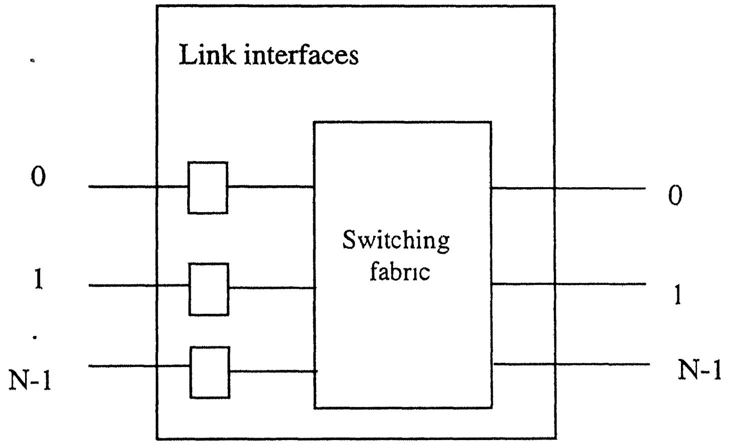


Fıg 3.1: Basic architecture of photonic switch based on FLM switching fabric

Power of the incoming signal to the swıtch is small, as the signal has to travel long

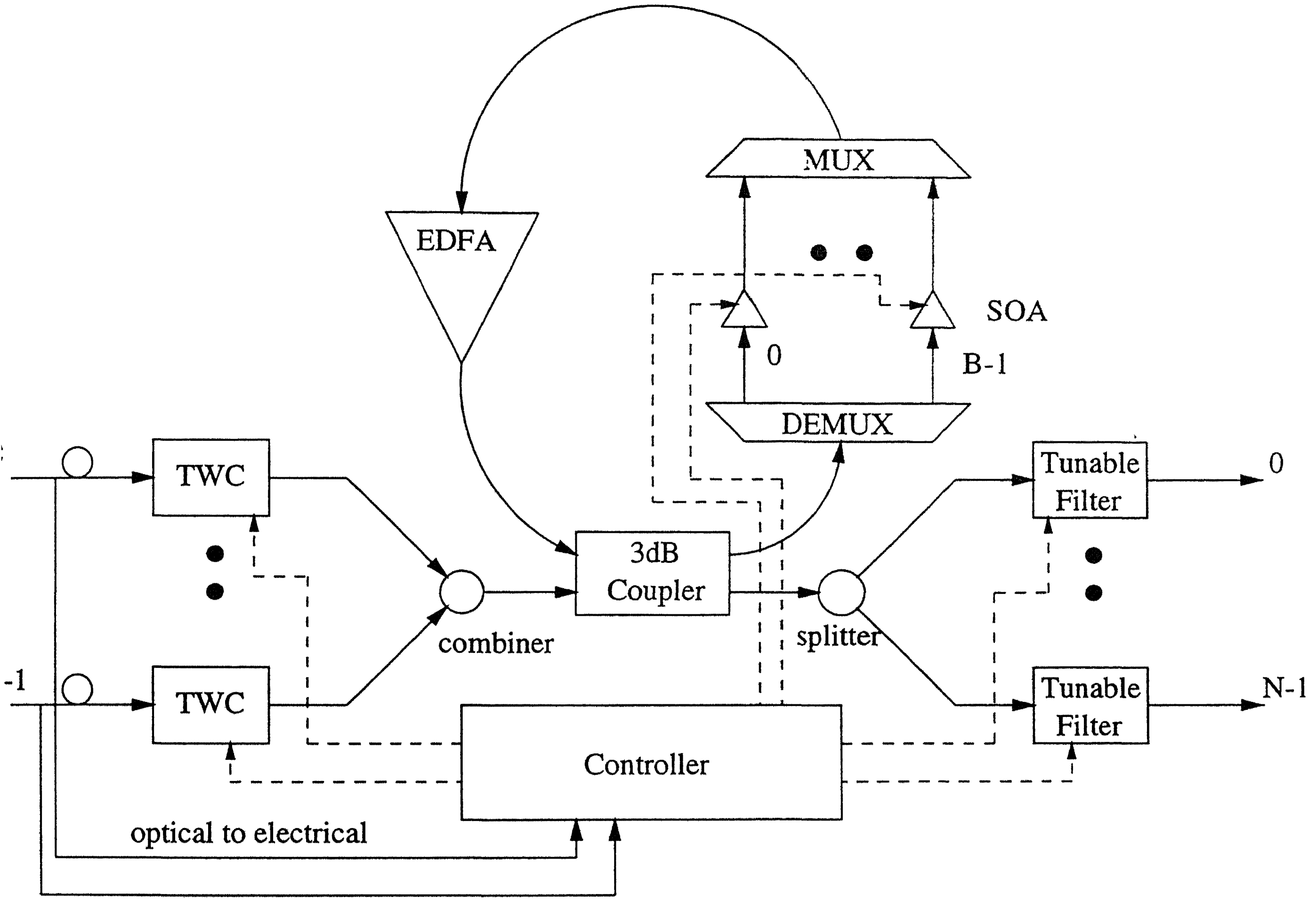


Fig 3.2 : Switch fabric architecture of the all-optical switch based on fiber loop memory

distances. So there is need to regenerate signal. This is done by link interfaces. There is need to synchronize different packets coming at the inputs of the switch, as switching fabric operations assumes packets at its inputs to be synchronous. This synchronization is also performed by link interfaces.

3.2 Switch fabric Architecture

The switching fabric architecture is shown in the fig 3.2. The multiwavelength fiber loop memory in the switching fabric operates in a WDM (wavelength division multiplexed) regime. The fabric shown in figure 3.2, can be considered as basic matrix which can be used to build large dimensional in multi stage switch configuration. The basic matrix is characterized by *N* input and *N* output ports. The fiber loop length is equal to one cell period. The switching fabric consists of four functional blocks.

- The cell-encoder block: incorporating *N* fast tunable optical wavelength converters.
- The buffering block: multiwavelength fiber loop buffer (memory) which is used in case of contention.
- The output block: consists of tunable filters.
- The electronic control block: this block implements the switching algorithm.

3.3 Operation of the switch

Assumptions

- Packets coming at the switch input are of fixed size.
- The operation of the switch fabric [2,6] is synchronous on packet period basis. (synchronization of packets is performed by link processor unit)
- There are delay elements at the inputs of the switch fabric, which provide delay so that processing of headers of incoming packets by the electronic controller is complete before the packet enter encoder block.
- The switch uses (*B*+*N*) wavelengths. Where '*B*' number of wavelengths corresponding to '*B*' SOAs in fiber loop, and '*N*' number of wavelengths are used in the case of direct transmission to the output .

- All-optical wavelength converters at the inputs and tunable filters at the outputs of the switch can be tuned to any of the (B+N) wavelengths instantaneously (ideally within zero time).

If there is only one packet for a particular destination from inputs (assume there is no packet in buffer for this output), electronic controller after processing the header of the incoming packet as discussed in Chapter 2, will tune the all-optical wavelength converter at the particular input and tunable filter at the output to one of the N wavelengths used for direct transmission in that time slot. So the packet will reach at all the outputs, but will be received by only one particular output for which it is destined.

In case of contention i.e., if there are more than one packet for some particular destination in the same slot (assuming that there are no packets for same output in buffers), one of the incoming packets is selected for direct transmission by tuning optical wavelength converter at that input and the tunable filter at the particular output to one of the N wavelengths used for direct transmission (controller has to make sure that no other input output pair involved in direct transmission is using the same wavelength in that time slot). Wavelength of each of the remaining packets is tuned to one of the free wavelengths (i.e., at which no packet is tuned in this time slot) in the fiber loop memory, if sufficient free wavelengths are available. Otherwise, controller will drop the packet. One implementation that is possible for dropping is to simply tune the particular packet to be dropped to one additional frequency at the output (This will increase the number of wavelengths in the switch to B+N+1). Thus the packets in fiber loop memory are kept recirculating (stored) in the WDM loop memory by activating the corresponding SOA in case it is required to be delayed. The switch gets the name from the fact that it will employ multiple wavelengths in the fiber loop to store the packets.

At the input of the memory loop, half of the power enters the loop and half goes toward the outputs through the passive coupler, so that when the contention is resolved, the packet is routed to the destination link simply by properly tuning the corresponding output tunable filter to the corresponding wavelength in the fiber loop. At the same time, the SOA in the loop is turned off to erase the packet in the memory. We can not erase and write into the same buffer (i.e., wavelength in the fiber loop) in the same time slot. The reason is explained in figure 3.3.

We assume that a packet is saved in T_{k-1} or any previous time slot in a SOA in the fiber loop memory, as there was more than one packet for some particular destination. In time slot T_k the packet saved in the particular SOA is selected for transmission. As soon as the slot in which packet is read at output starts controller will turn off the SOA i.e., at $T_k+\Delta t$ ($\Delta t \to 0$).

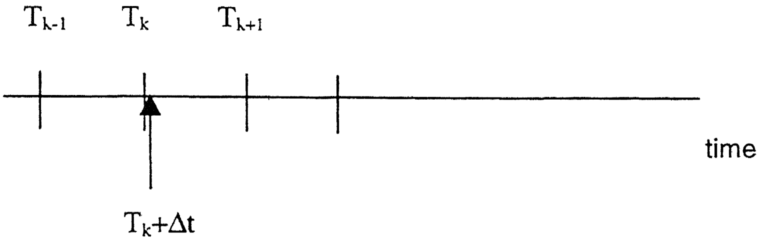


Fig 3.3 SOA in the fiber loop memory is turned off immediately the packet in it is selected for transmission

This causes the simultaneous erasure as well as reading of packet. As SOA remains off in the time slot T_k new packet can not be stored on this wavelength.

3.4 Description of components in the switch

The main components in the switch are 3-dB directional coupler, 1×*N* splitter and *N*×1 combiner (which can be made using *N*-1 2×2 3-dB couplers in multistage configuration), multiplexer/demultiplexer, SOA (semiconductor optical amplifiers), EDFA (Erbium doped fiber amplifier), tunable wavelength converters and tunable filters.

3.4.1 3-dB directional coupler

A directional coupler is used to combine and split the signals in an optical network. A 2×2 coupler consists of two input ports and two output ports, and is shown below in figure 3.4.

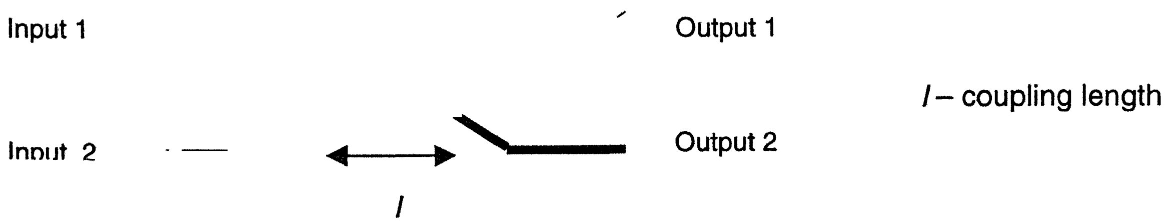


Fig 3.4: A directional coupler

One possible construction for a directional coupler is to fuse two fibers together in middle; another possibility is to fabricate it using waveguides in integrated optics. By a careful design, a coupler can be made wavelength independent over a usefully wide range. Such a coupler, shown in figure 3.4, takes a fraction α of the power from input 1 and places it on output 1 and the remaining fraction 1-α on output 2. Likewise, a fraction 1-α of the power from input 2 is distributed to output 1 and the remaining power to output 2. A coupler can be used as a power splitter if the coupling length, *l* in the figure 3.4, is adjusted such that half the power from each appears at each output. Such a coupler is called a 3-dB directional coupler.

3.4.2 Splitter and Combiner

A star coupler is a natural generalization of the 3-dB 2×2 coupler. In this switch a structure made using 3-dB 2×2 couplers is used at the input as combiner and at output as splitter. In general a *N*×1 combiner or 1×*N* splitter (where $N = 2^k$, k=1,2,..) can be realized using *N*-1 3-dB 2×2 couplers. For example an 8×1 combiner is shown in figure 3.5. It is realized from 7 3-dB couplers. Power of the signal after combining reduces to (1/*N*)th of that of at the input of combiner for an *N*×1 combiner.

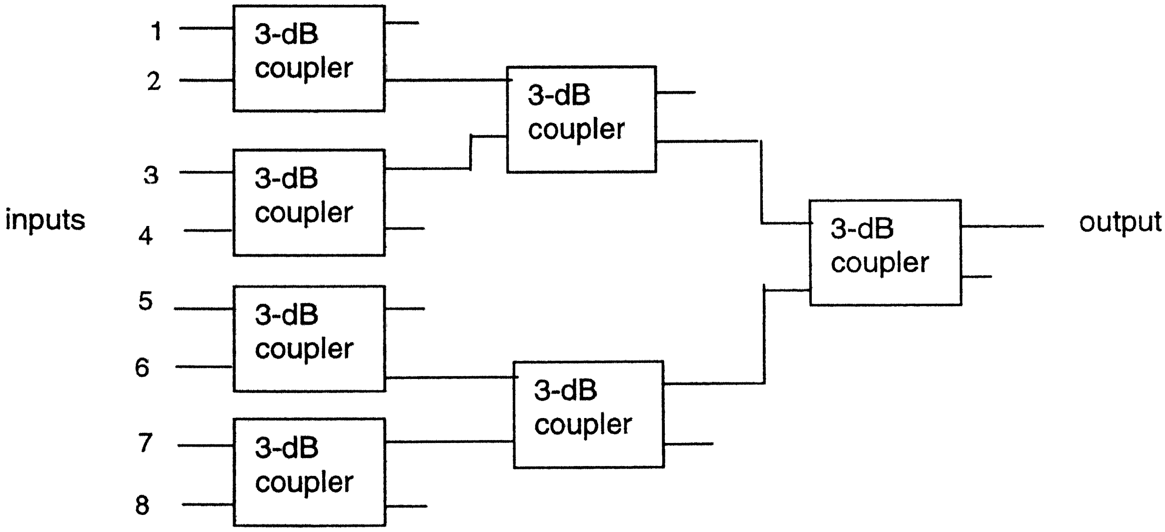


Fig 3.5: Star coupler as combiner

The same device acts as splitter (1×*N*) if input becomes output and output become input respectively. In this case also power at the output of splitter reduces to (1/*N*)th of that at the input.

3.4.3 Multiplexers/Demultiplexers

Function of a multiplexer is in general to combine many signals at its inputs to a signal at output as shown in figure 3.6. This can be realized as a wavelength flattened combiner in optical communications using WDM.

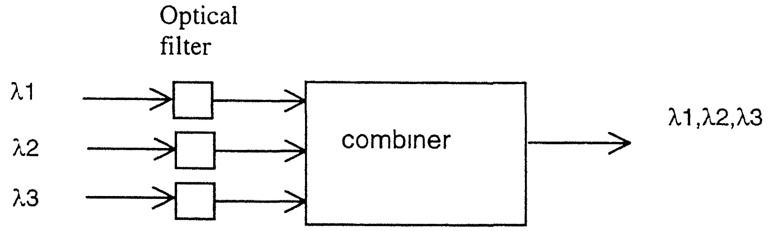


Fig 3.6: A WDM multiplexer

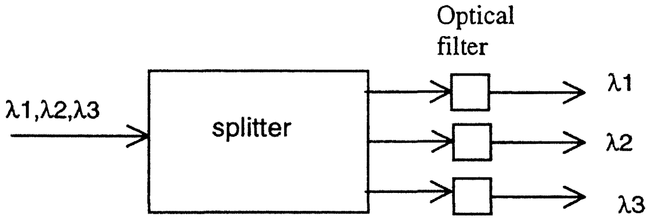


Fig 3.7: A WDM demultiplexer

Function of a demultiplexer in general is to separate the signal on different wavelengths in same fiber and send them to separate output fibers. This can be realized in general as a splitter followed by filters (figure 3.7). The outputs are then amplified by using SOAs in the fiber loop. SOAs acts as gates in the loop and can also be used to block the signals.

3.4.4 Wavelength converters

A wavelength converter is a device that converts incoming data from one incoming wavelength to another outgoing wavelength. Wavelength converters are useful components in WDM networks for three major reasons. First, data may enter the network at a wavelength that is not suitable for use within the network. Second, wavelength converters may be needed within the network to improve the utilization of the available wavelengths on the network links. Finally, wavelength converters may be needed as boundaries between different networks are managed by different entities and these entities do not coordinate the allocation of wavelengths in their networks. Wavelength converters can be classified based on the range of wavelengths that they can handle at their ınputs and outputs. A fixed-ınput, fixed-output device always takes ın a fixed-input wavelength and converts it to a fıxed wavelength. A variable-input, fixed-output device takes in a variety of wavelengths but always converts the input signal to a fixed-output wavelength. A fixed-input, varıable-output device does the opposite functıon Finally, a variable-input, variable-output device can convert any input wavelength to any output wavelength. There are three fundamental ways of achieving wavelength conversion (a) optoelectronic, (b) optical gatıng, and (c) wave mixing [11].

3.4.5 Tunable filters

A variety of optical filters are available. Their key characteristics for use in systems are the following.

1. Good optical filters should have low insertion losses. The insertion loss is the input-to-output loss of the device.
2. The loss should be independent of the state of polarization of the input signals. The state of polarization varies randomly with time in most systems, and if the filter has a polarization-dependent loss, the output power will vary with time as well, which is undesirable.
3. Time required for tuning must be very small (of the order of nano seconds).
4. Signal at the input of filter should not be reflected back.
5. The passband of a filter should be insensitive to variations in ambient conditions e.g., temperature.

3.4.6 SOA (Semiconductor Optical Amplifier), EDFA (Erbium-Doped Fiber Amplifier)

Amplifiers are required in the fiber loop because signal power decreases in the loop, as signal has to pass through demultiplexer, multiplexer (which are generally realized using 3-dB couplers as mentioned above) and 3-dB coupler in which power level decreases. If we are employing *B* buffers in the fiber loop and multiplexer, demultiplexer are realized in the loop as mentioned from 3-dB couplers then without amplifiers signal power level reduces to $1/2B^2$ after first rotation. In successive rotations this means very small power, and consequently signal will be lost. In order to increase the possible number of recirculations one has to employ amplifiers in the fiber loop as shown in figure 3.2.

Before going to brief description of SOA and EDFA two important physical phenomenon need to be discussed.

Stimulated Emission:

In SOA and EDFA, the key physical phenomenon behind signal amplification is stimulated emission of radiation by atoms. According to the principles of quantum mechanics, any physical system (for example, an atom) is found in one of a discrete number of energy levels. Accordingly, consider an atom and two of its energy levels, E_1 and E_2, with $E_2>E_1$. A photon whose frequency f_c satisfies $hf_c=E_2-E_1$ induces transitions of atoms between the energy levels E_1 and E_2. Here, h is Planck's constant (6.63×10^{-34} Js). This process is depicted in figure 3.8.

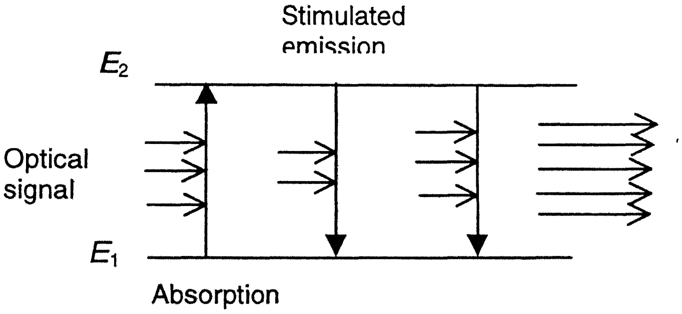


Fig 3.8: Stimulated emission and absorption in an atomic system with two, energy levels.

Both kinds of transitions, $E_1 \rightarrow E_2$ and $E_2 \rightarrow E_1$, occur. $E_1 \rightarrow E_2$ transitions are accompanied by absorption of photons from the incident beam. $E_2 \rightarrow E_1$ transitions are accompanied by the emission of photons of energy hf_c, the same energy as that of incident photons. This emission process is termed ***stimulated emission*** to distinguish it from another kind of emission called ***spontaneous emission***, which will be discussed later. The simulation emission results in photon, which is coherent in phase and direction with incident photon. If stimulated emission dominates over absorption-that is, the incident signal causes more $E_2 \rightarrow E_1$ transitions than $E_1 \rightarrow E_2$ transitions then one will have a net increase in the number of photons of energy hf_c and amplification of the signal will result. Otherwise, signal will be attenuated. It follows from the theory of quantum mechanics that the rate of $E_1 \rightarrow E_2$ transitions per atom equals the rate of the $E_2 \rightarrow E_1$

transitions per atom. Let this common rate be denoted by r. If the populations (number of atoms) in the energy levels E_1 and E_2 are N_1 and N_2, respectively, we have a net increase in power of $(N_2 - N_1)rhf_c$. Clearly, for amplification to occur, this must be positive, that is, $N_2 > N_1$. This condition is known as population inversion [11]. Population inversion can be achieved by supplying additional energy in a suitable form to the pump the electrons to the higher energy level. This additional energy can be in optical form or electrical form.

Spontaneous Emission:

Consider again the atomic system with the two energy levels discussed earlier. Independent of any external radiation that may be present, atoms in energy level E_2 transit to the lower energy level E_1' emitting a photon of energy hf_c. The spontaneous emission rate per atom from level E_2 to level E_1 is a characteristic of the system, and its reciprocal, denoted by τ_{21}, is called *spontaneous emission lifetime*. Thus if there are N_2 atoms in level E_2, the rate of spontaneous emission is N_2/τ_{21}, and the spontaneous emission power is hf_cN_2/τ_{21}. The spontaneous emission process does not contribute to the gain of the amplifier. Although the emitted photons have the same energy hf_c as the incident optical signal, they are emitted in random directions, polarizations, and phase. This is unlike the stimulated emission process where the emitted photons not only have the same energy as the incident photons but also the same direction of propagation, phase, and polarization. This phenomenon is usually described by saying that the stimulated emission process is coherent, whereas the spontaneous emission process is incoherent. The amplifier treats spontaneous emission radiation as another electromagnetic field at the frequency hf_c and the spontaneous emission also gets amplified, in addition to the incident optical signal. The *amplified spontaneous emission* (ASE) appears as noise at the output of the amplifier.

EDFA (Erbium-Doped Fiber Amplifier)

An doped fiber amplifier (DFA) schematic is shown in figure 3.9. It consists of a length of silica fiber whose core is doped with the rare earth elements. One of the commonly used dopant is Er^{3+} and results in EDFA [11]. This fiber is pumped using a pump signal from laser. So at the input end of doped fiber is a wavelength selective coupler which is used to combine the output of the pump laser with the input signal. At the output, another coupler separates the amplified signal from any remaining pump signal power. Usually an isolator is used at the input and/or output of amplifier to prevent reflections from going back to amplifier into the amplifier. A combination of several factors has made the EDFA the amplifier of choice in today's optical communication systems. This includes (a) the availability of compact and reliable high-power semiconductor pump lasers, (b) the fact that it is an all-fiber device, making it polarization independent. Further it can be splited with the transmission medium resulting in much smaller coupling losses, (c) the simplicity of the device, and (d) the fact that it introduces no crosstalk when amplifying WDM signals

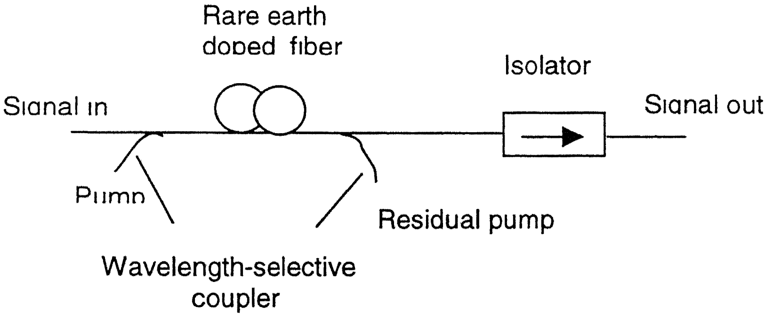


Fig 3.9: A doped fiber amplifier

SOA (Semiconductor Optical Amplifier)

The SOA is essentially a *pn*- junction. The depletion layer that is formed at the junction acts as the active region. Light is amplified through stimulated emission when it propagates through the active region. For an amplifier, the two ends of the active region are given an antireflection coating to eliminate ripples in the amplifier gain as a function of wavelength. Semiconductor optical amplifiers differ from EDFAs in the manner in which the population inversion is achieved [11]. In practice, bandwidths of the order of 100 nm can be achieved with SOAs. This is much larger than what is achievable with EDFAs. Signals in the 1.3 and 1.5 μm bands can even be simultaneously amplified using SOAs. Nevertheless, EDFAs are widely preferred to SOAs for several reasons. The main reason is that SOAs introduce severe crosstalk when they are used in WDM systems [11]. The gains and output powers achievable with EDFAs are higher. The coupling losses and polarization losses are also lower with EDFAs since amplifier is also a fiber. Finally, the SOAs require very high quality antireflective coatings on its faces, which is not easy to achieve. In the switch fabric shown in figure 3.2, SOAs are used to amplify signals at particular wavelengths and EDFA are used to amplify the WDM signal.

3.5 Concept of 'recirculation limit'

Consider a fiber loop with a single SOA in it as shown in figure 3.10.

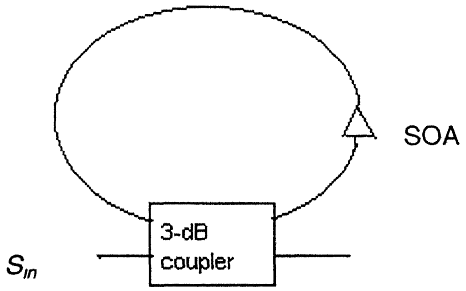


Fig 3.10: A fiber loop with SOA.

Consider initially S_{in} is the signal entering in to the loop. After first rotation total power at the SOA output will be $GS_{in}/2 + N_1$. Where, N_1 will be the ASE (Amplified Spontaneous Emission) noise generated in the SOA (the reason why it is generated is explained in 3.4.6), and G is the gain of the SOA. G is given by

$$G = G_0[e^{-(G-1)P_{in}/P_{sat}}] \qquad (3.1)$$

Where, G – gain of the SOA

G_0 – unsaturated gain of the SOA

P_{in}– input power for SOA (initially it is $S_{in}/2$)

P_{sat} – saturation power level of the SOA

After first rotation when signal enters second time in to the loop total power at the input of the SOA is ½ $(GS_{in}/2+N_1)$, and output power at the SOA is $G/2\times(GS_{in}/2+N_1) + N_2$. Where N_2 is the ASE noise added by the amplifier in the second recirculation. Hence

ASE noise gets accumulated in each recirculation within the loop. After few number of rotations S/N ratio decreases so much that we can not identify the required signal at the output. Hence there exists a limit on the number of recirculations possible. This is termed as 'recirculation limit'.

Chapter 4

Switching strategies for the switch

In this chapter a model for the switching fabric of the photonic switch based on mutiwavelength fiber loop memory is presented. Then the two switching strategies for the switch which has to be performed by the high-speed electronic controller (as there exists no optical processors) in the switch have been presented.

4.1 Model of the switching fabric

We can model the above switching fabric as a *N*×*N* shared buffer switch to which packets are coming synchronously because, the wavelengths in the fiber loop are shared by each output to save contending packets, with the following constraints

1) We can not erase and write in a buffer position in the same slot, which is possible in electronic shared buffer architectures.
2) There exists a limit on the max delay that can be provided by the switch. This limit comes from the fact that there exists a limit on number of recirculations possible for a packet.

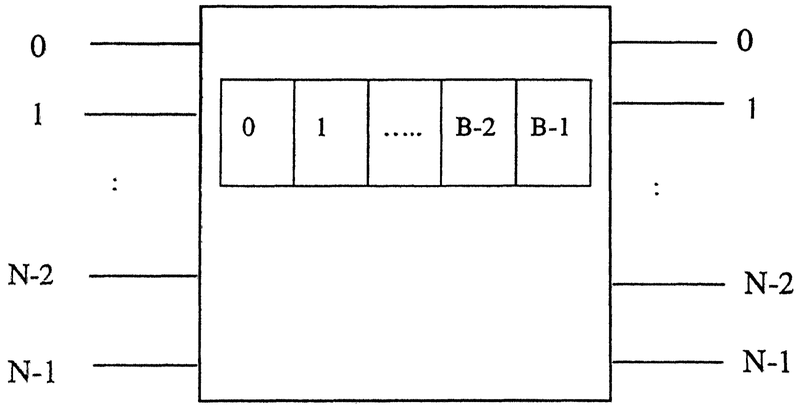


Fig 4.1: model of switching fabric for the photonic switch based on mutiwavelength fiber loop memory as a shred buffer switch with some constraints.

4.1.1 Traffic model

Here, we refer to the traffic as seen by the input ports of the switching fabric. The traffic model is described by two random processes. The first is that governs the arrival of packets in each time slot. The second process describes the distribution by which arriving packets choose their destination ports. In this thesis we assume traffic to be uniform. In this traffic model, packets arrive at each input ports of the switch according to independent and identically distributed Bernoulli processes, each with parameter p ($0 < p \leq 1$). In other words, at an input port in a given time slot, a packet arrives with probability p, and there is no arriving packet with probability $1-p$. Thus, p represents the input load or the arrival rate to each input port of the switching fabric. An incoming packet chooses its destination uniformly among all N output ports, and independently from all other requests i.e., it chooses a particular output with probability $1/N$. This traffic model is some times referred to as the independent uniform traffic model or simply the random traffic model. This assumption can be justified because it was observed that the traffic arriving at the switching nodes is less bursty than the traffic arriving at user nodes, due to inherent smoothing that takes place when bursty packet streams are queued and then released at a given rate (the link service rate) to the network. Furthermore, it was observed that the subsequent stages cause the traffic to become even less bursty, making the uniform traffic assumption closer to reality [14].

4.1.2 Modeling the 'recirculation limit'

The switch can not provide delay more than recirculation limit. Thus there exists a limit on the delay provided by the switch to packets going for any particular destination. We will indicate this maximum delay (i.e., recirculation limit) with b in our model. We are assuming that $B < Nb$, as it is not possible to use more number of wavelengths in the fiber loop memory due to noise limitations. If $B \geq Nb$, then this reduces to the simple case of output queueing structure with output queue length for any output not exceeding b, the recirculation limit and employing common buffers (B in number) for storing packets in the case of contention. Thus the switch can be modeled (assuming $B < Nb$) on overall as a shared buffered switch with constraint on maximum logical queue length for any particular output (i.e., logical queue length of the particular output can not exceed b, the recirculation limit) and we can not erase and write (or vice versa) in a buffer position in the same time slot.

4.2 'Drop inside' switching strategy

In general switching strategies required in the switch has to utilize buffer space efficiently. Here in this switch this issue is more important than in normal circumstances, as we can not use more than certain number of wavelengths in the loop to save contending packets due to noise limitations.

In 'drop inside' switching strategy in the case of contention controller will check in every time slot whether there are sufficient number of available free wave lengths in the fiber loop to save contending packets for each of the output (one packet will be transmitted in the time slot if any, for each output). If there are not sufficient number of available free wavelengths in the fiber loop, controller will drop newly arrived packets by randomly selecting some. In this process it is made sure that there will be at least one packet if any, for transmission to each output. After dropping packets, all remaining packets, except those selected for transmission from newly arrived packets in this slot are saved by turning the SOAs in the buffer positions ON. Packets for any particular output can recirculate at most b recirculations (i.e., can be provided delay of at most b). If any packet has rotated more than b recirculations the signal power corresponding to the information bits in the packet reduces so much that receiver will not able to identify the signal (i.e., information bits of the packet), and such packets are erased from fiber loop memory (or dropped inside the loop). That is why we are calling this switching strategy as 'drop inside' strategy. In the beginning of each time slot controller will check the status of packets saved in the loop. If it identifies that any packet in the fiber loop has rotated for more than b recirculations, it will free the buffer by turning OFF the SOA corresponding to that buffer position in which packet is saved. The algorithm for this strategy is presented below and then explained.

Algorithm:

–All packets upon arrival in a slot are analyzed for their destination. For each output '*i*' number of new packets arrived in this slot are determined.

–All packets in buffer are analyzed and arranged in queues for each output '*i*' with packet having larger delay being ahead in queue.

–If delay for any packet in buffer is greater than *b* (recirculation limit) it is marked as dropped. But as the packet will be erased in current time slot, no new packet can be stored on this wavelength This wavelength becomes free in next time slot.

–For each output '*i*',

If there is packet in buffer, no packet from input is assigned direct wavelength.

If there is no packet in the buffer, one of the packet from input is assigned direct wavelength.

–Determine the number of free wavelengths on which packets can be stored. Find number of packets at inputs need to be stored.

While (number of packets need to be stored at inputs > free wavelengths)

Choose randomly one packet from input and drop it.

–Assign free wavelengths to all the remaining input packets to be stored.

–All the packets in buffers with maximum delay are transmitted and corresponding buffer positions are marked empty.

The above algorithm transcends to the following steps to compute average delay and probability of blocking. Before going to these steps variables used in them are defined.

'*sum*' = sum of all the packets to all '*i*'s arrived in all the slots,

'*dropped*' = total number of packets dropped in all the slots,

'*delay*' = sum of delays of transmitted packets in all slots,

'*totaltx* ' = the count for total transmitted packets.

'*pac_thisslot*' = number of packets arrived in that particular slot, '*bufreqd*' = number of buffers required in that particular time slot,

'*freebuf*' represents free buffer positions available for storage in that particular timeslot, '*dropinside*' is a variable representing the number of packets that need to be dropped in side buffer in the particular slot as these packets have rotated already *b* recirculations, '*posinuse*' is a variable saving the buffer positions in which packets have delay greater than *b* and will be dropped in current slot.

For each output *i* (*i*=0 to *N*-1, where *N* is number of input or output lines) there are two logical queues. (1) From buffer to *i*, represented by $W_{buf,i}$ of length $X_{buf,i}$ and (2) From newly arrived packets in that time slot to *i*, represented by $W_{new,i}$ of length $X_{new,i}$ as shown in the figure 4.2.

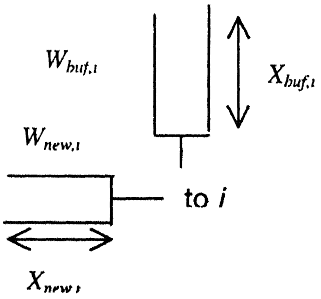


Fig 4.2: logical queues used in the algorithm.

'*bufreqd[i]*' represents number of buffers required to save packets contending for *i*, '*selected[i]*' represents buffer position or input port number from which packet will be selected for transmission for '*i*' th output.

Step 1: *sum*=0, *dropped*=0, *delay*=0, *totaltx*=0.

Step 2: Get packets from link-processor. *sum* = *sum* + *pac_thisslot*, set *dropinside*=0,

bufreqd = 0, *freebuf* = 0.

Step 3: Check if in any buffer position delay > *b*. If delay > *b* for any buffer position increment '*dropinside*' by 1 and note that buffer position in '*posinuse*'. Drop the packet from that buffer position and turnoff SOA. For each dropping increment '*dropped*' (i.e., *dropped* = *dropped* +*1*).

Step 4: Form logical queues from buffer to i, $W_{buf,i}$ of length $X_{buf,i}$ and from newly arrived packets in that time slot to i, represented by $W_{new,i}$ of length $X_{new,i}$.

bufreqd[i] = $X_{new,i}$, if $X_{buf,i} > 0$

= $X_{new,i}$ - 1 , if $X_{buf,i}$ = 0 and $X_{new,i}$ > 0, because one packet can use direct wavelength for transmission

= 0, otherwise

$$bufreqd = \sum_{i=0}^{N-1} bufreqd[i]\ , freebuf = B - \sum_{i=0}^{N-1} X_{buf,i} - \text{'}dropinside\text{'}.$$

Define $diff = bufreqd - freebuf$. If ($diff$>0) select some i such that [($X_{new,i}$ > 1) or ($X_{buf,i}$ > 0 and $X_{new,i}$ > 0)] randomly. Drop a packet randomly from queue $W_{new,i}$. *dropped* = *dropped* +1, $diff = diff - 1$. Repeat this process (i.e., selecting randomly i and dropping) till $diff$=0.

After this step $X_{new,i}$ becomes $X^1_{new,i}$ and $W_{new,i}$ becomes $W^1_{new,i}$

Step 5: *selected[i]* = randomly one from $W_{buf,i}$ with maximum delay, if $X_{buf,i} > 0$

= randomly one from $W^1_{new,i}$, if $X_{buf,i} = 0$ and $X^1_{new,i} > 0$

= N+1. (i.e., none)

if *selected[i]* != N+1, *delay* = *delay* + delay of *selected[i]*, *totaltx* = *totaltx* +1.

Step 6: Save remaining packets from $W^1_{new,i}$ (after selection) in to buffers by making sure that this buffer position is not one of '*posinuse*', for i =0 to N-1. Transmit *selected[i]*. Free the buffer positions from which packets are transmitted in this slot.

Step 7: Increment delay of packets within buffer by 1. Increment present timeslot by 1. If present timeslot is not equal to time slot count go to *Step* 2, else go to *Step* 8.

Step 8: Count parameters of interest

packet loss probability = dropped / sum.

average delay = delay/ totaltx.

In *Step* 1 '*sum*', '*dropped*', '*delay*', '*totaltx*' are initialized to zero.

In *Step* 2 Packets after synchronization at link processor will arrive to switching fabric at the beginning of timeslot. Arrival process is uniformly distributed. '*dropinside*' is initialized to 0. Controller will check in the beginning whether for any packet in the buffer the delay is greater than b and the packets are still required to provide delay. Controller will turn off the SOA corresponding to the particular buffer position and that buffer position is not available for storage as explained in section 3.3. So controller need to make sure that, the buffer position (which has been freed in this slot) should not be used in the same slot. So controller is saving the buffer positions which have been freed in this slot in a variable '*posinuse*'. Each time controller is dropping a packet inside it will be incrementing '*dropped*' by 1, '*dropinside*' by 1, and controller will note the corresponding buffer position in '*posinuse*'. This is done in *Step* 3.

In *Step* 4, controller first forms logical queues $W_{buf,i}$ of length $X_{buf,i}$ and $W_{new,i}$ of length $X_{new,i}$ as explained in fig 4.2 for each i. Buffers required in this slot to save contending packets is calculated. '*bufreqd[i]*' represents number of buffers required to save packets contending for i . This is equal to $X_{new,i}$ if, $X_{buf,i} > 0$. Reason is that, if there is any packet in the buffer that will be transmitted, in order to maintain FCFS (First Come First Serve) service. All the remaining packets for that destination have to be saved. If $X_{buf,i} = 0$ and $X_{new,i} > 0$, which means that there are only newly arrived packets, for destination i, then one of them will transmitted and remaining has to be saved. In this case '*bufreqd[i]*' equals $X_{new,i} - 1$. Other wise (i.e., $X_{buf,i} = 0$, $X_{new,i} = 0$), implies that there are no packets. Total number of buffers required are for all i's (i=0 to N-1) is '*bufreqd*' equals sum of '*bufreqd[i]*' over all i's. Free buffers in the slot are given by '$B - \sum_{i=0}^{N-1} X_{buf,i}$'. But controller has to make sure that it should not use the buffer positions of packets that are dropped in this slot in *Step* 3. So free buffers that are actually available for storage in the slot are given by '*freebuf*' equals 'B - $\sum_{i=0}^{N-1} X_{buf,i}$ - *dropinside*'. If packets to be saved exceeds than free buffer positions available, controller will drop '*diff*' (= *bufreqd* – *freebuf*) number of packets from newly arrived packets by randomly selecting some i, by making sure there exists at least one packet for transmission. The conditions which makes sure this is given as, select an i, such that $[(X_{new,i} > 1)$ or $(X_{buf,i} > 0$ and $X_{new,i} > 0)]$ randomly. From the queue $W_{new,i}$ packets are dropped randomly, till '*diff*' becomes 0 as in the algorithm. After this step $X_{new,i}$ becomes $X^1_{new,i}$ $(0 \leq X^1_{new,i} \leq X_{new,i})$, $W_{new,i}$ becomes $W^1_{new,i}$ (may be same as $W_{new,i}$ or some packets in this may have lost).

In *Step* 5, controller is selecting packets for transmission. Packets selected for transmission on i is from $W_{buf,i}$ with maximum delay(randomly) if $X_{buf,i} > 0$, else if $X_{buf,i} =$ 0 and $X^1_{new,i} > 0$ then *selected[i]* is randomly one from $W_{new,i}$. Other wise, there is no packet on that line (this is represented by the case *selected[i]*= N+1. Controller will add the delay of selected packet to delay, increments *totaltx*, if packet is selected on i. This is done for all i's.

In *Step* 6, remaining packets in $W^1_{new,i}$ are saved in buffer positions, by making sure that the buffer position selected is not in '*posinuse*'. Selected packets are transmitted. If selected packets are from buffers then corresponding buffer positions will be freed. Controller will increment delay for the packets saved in buffers before going for processing in next time slot. This is repeated. *Packet loss probability*, *average del*ay are calculated as explained in *Step*s 7, 8 of algorithm.

4.3 'Dropping at the inputs' switching strategy

In 'Dropping inside' strategy, in the case of contention, packets will enter in to the buffer when they finds free buffers even when there are already more than or equal to b (recirculation limit) number of packets in the buffer for the same destination for which this newly arrived packets are destined. If there are already b or more packets in the buffer for a particular destination there is no point in saving the newly arrived packets for same destination in to the buffers, as they will be dropped inside the loop after recirculating b number of timeslots. So in 'Dropping at the inputs' strategy controller has to drop packets destined for a particular output port at the inputs from newly arrived packets, if total number of packets (i.e., from buffer and from newly arrived) for the destination exceeds b+1 Where one represents the fact that one packet will be transmitted in this slot. This directly means that by using this switch a delay more than b (recirculation limit) can not be provided as explained in section 4.1. In 'Dropping at the inputs' strategy controller will make sure that total number of packets in the buffer for any particular destination will not exceed b at the end of any timeslot. This strategy is expected to use buffers efficiently compared to ' Dropping inside strategy'. Algorithm for this switching strategy is presented and then explained. The two switching strategies are compared in Chapter 6.

Algorithm:

–All packets upon arrival in a slot are analyzed for their destination. For each output 'i' number of new packets arrived in this slot are determined.

–All packets in buffer are analyzed and arranged in queues for each output 'i' with packet having larger delay being ahead in queue.

–If total number of packets (from buffers and newly arrived) for any destination 'i' is exceeding b+1, some of the newly arrived packets for that destination 'i' are dropped randomly, to limit total number of packets for 'i' to b+1.

–For each output 'i',

If there is packet in buffer, no packet from input is assigned direct wavelength.

If there is no packet in buffer, one of the packet from input is assigned direct wavelength.

–Determine the number of free wavelengths on which packets can be stored. Find number of packets at inputs need to be stored.

While (number of packets need to be stored at inputs > free wavelengths)

Choose randomly one packet from input and drop it.

–Assign all the remaining input packets to be stored free wavelengths.

–All the packets in buffers with maximum delay are transmitted and corresponding buffer positions are marked empty.

The above algorithm transcends to the following steps to compute average delay and probability of blocking. Before going to these steps variables used in them are defined.

'*sum*', '*dropped*', '*delay*', '*totaltx*', '*pac_this slot*', '*bufreqd*', '*freebuf*', '*bufreqd[i]*', '*selected[i]*', $W_{buf,i}$, $X_{buf,i}$, $W_{new,i}$, $X_{new,i}$ have same meaning as in 'drop inside' switching strategy. '*total$_i$*' represents total number of packets to '*i*' th destination in that particular time slot.

Step 1: *sum* =0, *dropped* =0, *delay* =0, *totaltx* = 0.

Step 2: Get packets from link-processor. *sum* = *sum* + *pac_this slot*, set *bufreqd* = 0, *freebuf* = 0.

Step 3: Form logical queues from buffer to *i*, $W_{buf,i}$ of length $X_{buf,i}$ and from newly arrived packets in that time slot to *i*, represented by $W_{new,i}$ of length $X_{new,i}$. Define $total_i = X_{buf,i} + X_{new,i}$. If ($total_i$ >[*b*+1]) drop $diff1 = total_i - [b+1]$ number of packets randomly from logical queue $W_{new,i}$. For each packet dropping increment '*dropped*' (i.e., *dropped* = *dropped* +1).

After this step $X_{new,i}$ becomes $X^1_{new,i}$ and $W_{new,i}$ becomes $W^1_{new,i}$. Do *Step* 3 for each *i* (i.e., *i*=0 to *N*-1).

Step 4: $bufreqd[i] = X^1_{new,i}$, if $X_{buf,i} > 0$

$= X^1_{new,i} - 1$, if $X_{buf,i} = 0$ and $X^1_{new,i} > 0$, since one packet will be using direct wavelength.

$= 0$, otherwise

$$bufreqd = \sum_{i=0}^{N-1} bufreqd[i]\ ,\ freebuf = B - \sum_{i=0}^{N-1} X_{buf,i}\ .$$

Define $diff = bufreqd - freebuf$. If ($diff$>0) select some i such that [($X^{1}_{new\ i}$ > 1) or ($X_{buf,i}$ > 0 and $X^{1}_{new,i}$ > 0)] randomly. Drop a packet randomly from queue $W^{1}_{new\ i}$. Increment '*dropped*' by 1 for each packet dropped and decrement '*diff*' by 1(i.e., $dropped = dropped + 1$, $diff = diff - 1$). Repeat this process (i.e., selecting randomly i and dropping) till $diff$=0.

After this step $X^{1}_{new,i}$ becomes $X^{2}_{new,i}$ and $W^{1}_{new,i}$ becomes $W^{2}_{new,i}$

Step 5: *selected[i]* = randomly one from $W_{buf,i}$ with maximum delay, if $X_{buf,\ i} > 0$

= randomly one from $W^{2}_{new,i}$, if $X_{buf,i} = 0$ and $X^{2}_{new,i} > 0$

= N+1. (i.e., none)

if *selected[i]* != N+1, *delay* = *delay* + delay of *selected[i]*, *totaltx* = *totaltx* +1.

Step 6: Save remaining packets from $W^{2}_{new,i}$ (after selection) in to buffers.

for i =0 to N-1, transmit *selected[i]*. Free the buffer positions from which packets are transmitted in this slot.

Step 7: Increment delay of packets within buffer by 1. Increment present timeslot by 1. If present timeslot is not equal to time slot count go to *Step* 2, else go to *Step* 8.

Step 8: Count parameters of interest

packet loss probability = dropped / sum.

average delay = delay/ totaltx.

In *Step* 1 '*sum*', '*dropped*', '*delay*', '*totaltx*' are initialized to zero.

In *Step* 2 Packets after synchronization at link processor will arrive to switching fabric at the beginning of timeslot. Arrival process is uniformly distributed.

In *Step* 3, controller will form logical queues as explained. In this strategy controller will now check whether total packets to *i*, '$total_i$' is greater than $b+1$ (one represents the fact that one packet will transmitted in this slot, and the switch can not provide delay more than *b*). If $total_i$ is greater than $b+1$ controller will drop $diff1 = total_i - [b+1]$ number of packets to limit total number of packets for *i* to $b+1$. So $X_{new,i}$ becomes $X^1_{new,i}$ $(0 \leq X^1_{new,i} \leq X_{new,i})$, $W_{new,i}$ becomes $W^1_{new,i}$ (may be same as $W_{new,i}$ or some packets in this queue may have lost).

Steps 4, 5 are same as in first strategy but $X_{new,i}$ replaced by $X^1_{new,i}$, $W_{new,i}$ replaced by $W^1_{new,i}$ and $X^1_{new,i}$ replaced by $X^2_{new,i}$, $W^1_{new,i}$ replaced by $W^2_{new,i}$ and requires no explanation.

In *Step* 6, controller will select free buffers and save packets in them. And selected packets in *Step* 5 are transmitted. The buffer positions from which packets are transmitted in this slot are freed.

Controller will increment delay for the packets saved in buffers before going for processing in next time slot. This is repeated. Packet loss probability, average delay are calculated as explained in *Step*s 7, 8 of algorithm.

Thus the main difference between 'dropinside' strategy and 'drop at the inputs' strategy is that how packets are dropped due to *b* limitation. In first strategy packets are dropped inside the loop after recirculating more than *b* times. In second strategy packets are dropped at the inputs if total number of packets for any output are exceeding *b*+1.

Chapter 5
Priority switching algorithm for the switch

We have discussed about priority requirements in packet switching networks in Chapter 1. In this chapter we propose a priority switching algorithm for the photonic switch based on multiwavelength fiber loop memory.

5.1 Issues in priority switching

Generally statistical multiplexing is used in packet switched networks as it provides more efficient available bandwidth utilization compared to fixed multiplexing [11]. Because of statistical multiplexing of a large number of sources in a packet switched network, congestion may arise in the network due to overlap of the peak rate of different sources in time. This could lead to the possibility of packets being lost inside the network. Different approaches have been proposed in the literature to meet QoS (Quality of Service) requirements in a packet switched network in the case of congestion in the switch buffer. The main QoS requirements deal with cell loss rate and transfer delay characteristics. A flexible way to provide different QoS is to provide some priority mechanism inside the network. Priorities can be (a) delay (or time) priorities (b) loss (or space) priorities. Delay priorities provide preferential service to some classes of traffic in order to control their end to end delay and delay variation. Loss (or space) priorities provide preferential access to buffer space. We will consider only loss priorities. Recently some investigations of various space priority mechanisms have appeared in the literature [13,15].

Generally to ease the decision regarding which packet to be dropped in the case of congestion, the packets are classified as high priority (or Class 1) which are loss sensitive, and low priority (or Class 2) which are loss insensitive. The pushout scheme [13] is a preemptive technique where an arriving Class 1 (high priority) packet can push out (over write) a Class 2 (low priority) packet with minimum delay if it finds the buffer full. This requires two operations to be performed in the same slot (a) removing (dropping) the low priority packet from buffer position and (b) keeping (writing) newly arrived high priority packet in that particular buffer position.

But in the photonic packet switch based on multiwavelength fiber loop memory in the same slot we can not drop packet from a particular buffer position and write a packet in to it as explained in section 3.3 (figure 3.3). This means 'pushout priority scheme' can *not* be *employed* for the photonic switch based on multiwavelength fiber loop memory. This implies one needs a different priority switching strategy so that high priority packet (which is loss sensitive) will suffer relatively low losses compared to low priority packets (which are loss insensitive).

5.2 Priority switching algorithm for the photonic packet switch based on multiwavelength fiber loop memory

In this algorithm a newly arrived high priority packet finds if there are already b (where b is the maximum queue length that is permitted for any particular output) number of packets in the buffer for the same destination to which it wants to go, then a low priority packet can not be removed from the buffer. Consequently a high priority packet can not replace low priority packet as done in 'pushout' strategy. So the newly arrived high priority packet will try to find position in buffer, then it will pushout low priority packet from buffer to limit total number of packets to b for that output. The algorithm is presented below.

Algorithm:

There exist basically three limitations in the switch. Which are

1. In the same time slot it is not possible to drop (or erase) and write a packet in to a buffer position.
2. It is not possible to provide delay of more than b (the recirculation limit) time slots using the switch.
3. Total number of buffers B are such that $B < Nb$. In the priority algorithm it is made sure that all the limitations are reached as explained below.

–All packets upon arrival in a slot are analyzed for their destination, priority. For each output '*i*' number of new packets arrived with high priority and low priority in this slot are determined.

–All packets in buffer are analyzed for destination, priority and arranged in queues for each output '*i*', with packet having high priority, larger delay being ahead in queue.

–If delay for any packet in buffer is greater than *b* (recirculation limit), it is marked as dropped But as the packet will be erased in current time slot, no new packet can be stored on this wavelength. This wavelength becomes free in next time slot.

–For each output '*i*',

If there is packet in buffer and that is having high priority, no packet from input is assigned direct wavelength

If there is no high priority packet in buffer and there are one are more high priority packets at inputs, one of the high priority packet from input is assigned direct wavelength.

If there is no high priority packet in buffer and there are no high priority packets at inputs and there are low priority packets in buffer, no packet from input is assigned direct wavelength.

If there is no packet in buffer and no high priority packets at inputs, one of the low priority packets from input is assigned direct wavelength.

–If there is at least one high priority packet for destination '*i*' and sum of newly arrived packets and high priority packets from buffer to the destination is exceeding *b*+1, limit the sum of packets mentioned above to *b*+1 by first dropping newly arrived low priority packets till either the limit is achieved or there are no more low priority packets for that destination. If there are no more low priority packets for that destination still limit has not been achieved then drop newly arrived high priority packets to limit total number of packets for the destination to *b*+1.

–Check whether the number of the remaining packets (from buffer, newly arrived) for any destination '*i*' is greater than b+1. If total number of packets for the destination are greater than b+1, then try to limit the total number of packets for the destination to b+1, select randomly one of the newly arrived low priority packets and drop it. This is done till limit is reached or there are no more newly arrived low priority packets.

–Determine the number of free wavelengths on which packets can be stored. Find number of packets at inputs need to be stored.

```
While (number of packets need to be stored at inputs > free wavelengths){

        Choose randomly one of the low priority packet from input and drop it.

    If there are no more low priority packets for any destination then only choose
randomly one of the high priority packets at input and drop it.

}
```

–Assign free wavelengths to all the remaining high priority packets at input to be stored.

-Check whether the number of the remaining packets (from buffer, newly arrived) for any destination '*i*' is greater than b+1. If total number of packets for the destination are greater than b+1, then to limit the total number of packets for the destination to b+1, select randomly one of the newly arrived low priority packets and drop it. If there are no more newly arrived low priority packets for that destination and still total number of packets for that destination are exceeding b+1, then erase some of the stored low priority packets in buffer which have low delay, till the limit is achieved

–Assign all the remaining low priority packets at input to be stored free wavelengths.

–All the packets in buffers with maximum delay taking consideration in to priority are transmitted and corresponding buffer positions are marked empty.

The above algorithm transcends to the following steps to compute average delay and probability of blocking for both classes of arrivals. Before going to these steps variables used in them are defined.

highsum, *lowsum* represents total number of high priority packets, low priority packets respectively to all *i*'s arrived in all the slots.

highdropped, *lowdropped* represents respectively the total number of high priority , low priority packets dropped in all the slots.

highdelay, *lowdelay* represents respectively the sum of delays of transmitted high priority, low priority packets in all slots.

hightotaltx, *lowtotaltx* represents respectively the count for total transmitted high priority, low priority packets.

'*bufreqd*' represents number of buffers required in that particular time slot.

'*freebuf*' represents free buffer positions available for storage in that particular timeslot.

'*dropinside*' represents the number of packets that need to be dropped in side buffer in that particular slot (as these packets have rotated already *b* recirculations, and needs to be saved for some more timeslots).

For each output *i* (*i*=0 to *N*-1, where *N* is number of input or output lines) there are four logical queues. (1) From buffer to *i*th output high priority packets, represented by $W_{bufhigh,i}$ of length $X_{bufhigh,i}$, (2) From buffer to *i*th output low priority packets, represented by $W_{buflow,i}$ of length $X_{buflow,i}$,

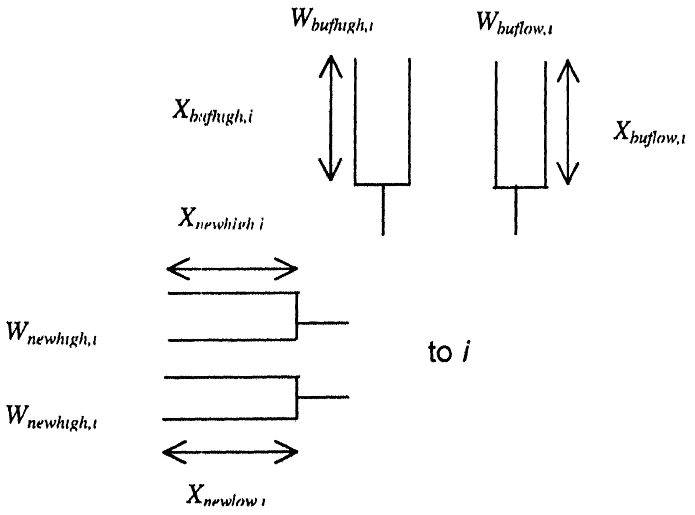


Fig 5.1: Logical queues used in algorithm

(3) From newly arrived high priority packets in that time slot to *i*th output, represented by $W_{newhigh,i}$ of length $X_{newhigh,i}$, (4) From newly arrived low priority packets in that time slot to *i*th output, represented by $W_{newlow,i}$ of length $X_{newlow,i}$ as shown in the figure 5.1. '*posinuse*' is a variable saving the buffer positions in which packets have delay greater than *b* and will be dropped in this slot.

'*selectedlow*' represents number of total low priority packets that will be transmitted directly in that particular time slot.

'*bufreqd[i]*' represents number of buffers required to save packets contending for *i*, '*selected[i]*' represents buffer position or input port number from which packet will be selected for transmission for '*i*' th output.

Step 1: *highsum*=0, *highdropped*=0,*highdelay*=0,*hightotaltx*=0,*lowsum*=0,*lowdropped*=0, *lowdelay*=0, *lowtotaltx*=0.

Step 2: Get packets from link-processor. Increment *highsum* by 1 if packet coming is of high priority and *lowsum* if packet coming is of low priority. Set *dropinside*=0, *bufreqd*=0, *freebuf* =0.

Step 3: Check if in any buffer position delay > *b*. If delay > *b* for any buffer position increment '*dropinside*' by 1 and note that buffer position in '*posinuse*'. Drop the packet from that buffer position and turnoff SOA. For each dropping increment '*highdropped*' (i.e., *highdropped* = *highdropped* +1) if the packet dropped is of high priority, else if packet dropped is of low priority increment '*lowdropped*'.

Step 4: form logical queues from buffer to *i*, $W_{bufhigh,i}$ of length $X_{bufhigh,i}$, $W_{buflow,i}$ of length $X_{buflow,i}$ and from newly arrived packets in that time slot to *i*, represented by $W_{newhigh,i}$ of length $X_{newhigh,i}$, $W_{newlow,i}$ of length $X_{newlow,i}$. Define $total_i = X_{bufhigh,i} + X_{newhigh,i} + X_{newlow,i}$.

If ($total_i$ >*b*+1)

Define *diff1* = $total_i$ – [*b*+1],

while(*diff1* > 0) drop packets

(a) randomly from logical queue $W_{newlow,i}$, if($X_{newlow,i} > 0$).

For each packet dropping increment '*lowdropped*' by 1 (i.e , *lowdropped = lowdropped* +1), decrement $X_{newlow,i}$ (i e., $X_{newlow,i} = X_{newlow,i} - 1$), decrement '*diff1*'(i.e.,$diff1 = diff1 - 1$).

(b) randomly from logical queue $W_{newhigh,i}$,if($X_{newlow,i} = 0$).

For each packet dropping increment '*highdropped*' by 1 (i.e., *highdropped* = *highdropped* +1), decrement $X_{newhigh,i}$ (i.e., $X_{newhigh,i} = X_{newhigh,i} - 1$), decrement *diff1* (i.e.,$diff1 = diff1 - 1$).

After this step $X_{newlow,i}$ becomes $X^{1}_{newlow\,i}$ and $W_{newlow,i}$ becomes $W^{1}_{newlow,i}$ and $X_{newhigh,i}$ becomes $X^{1}_{newhigh,i}$ and $W_{newhigh,i}$ becomes $W^{1}_{newhigh,i}$. Do Step 4 for each i (i.e., *i*=0 to *N*-1).

Step 5:

Define $grandtotalfirst[i] = X_{bufhigh,i} + X^{1}_{newhigh,i} + X_{bufhigh\,i} + X^{1}_{newlow,i}$.

If ($grandtotalfirst[i] > [b+1]$) define $diff2 = grandtotal[i] - [b+1]$.

while ($diff2>0$ and $X^{1}_{newlow,i} > 0$) {

Drop packets from $W^{1}_{newlow,i}$. Increment '*lowdropped*' by 1 (i.e., *lowdropped =lowdropped* +1) for each dropping packet. Decrement $X^{1}_{newlow,i}$ by 1 and *diff2* by 1.

}

After this process $X^{1}_{newlow,i}$ becomes $X^{2}_{newlow,i}$, $W^{1}_{newlow,i}$ becomes $W^{2}_{newlow,i}$.

Step 6 : $bufreqd[i] = X^{1}_{newhigh,i} + X^{2}_{newlow,i}$, if ($X_{bufhigh,i} > 0$)

$$= X^{1}_{newhigh,i} - 1 + X^{2}_{newlow,i} \text{, if } (X_{bufhigh,i} = 0 \text{ and } X^{1}_{newhigh,i} > 0)$$

$= X^2_{newlow,i}$, if ($X_{bufhigh,i} = 0$ and $X^1_{newhigh,i} = 0$ and $X_{buflow,i} > 0$)

$= X^2_{newlow,i}$ - 1, if ($X_{bufhigh,i} = 0$ and $X^1_{newhigh\,i} = 0$ and $X_{buflow,i} = 0$ and $X^2_{newlow\,i} > 0$)

= 0, otherwise.

Set a variable *selectedlow* = 0.

Increment '*selectedlow*' by 1 if *bufreqd[i]* = $X^2_{newlow,i}$ - 1.

$$bufreqd = \sum_{i=0}^{N-1} bufreqd[i]\,, freebuf = B - \sum_{i=0}^{N-1}(X_{bufhigh,i} + X_{buflow,i}) - dropinside.$$

Define $diff = bufreqd - freebuf$, and $tmp = \sum_{i=0}^{N-1} X^2_{newlow,i}$ - *selectedlow*.

while (*diff*>0)

(a) while (*tmp* > 0)

select some *i* such that [($X^2_{newlow,i} > 1$) or {($X_{bufhigh,i} > 0$ or $X^1_{newhigh,i} > 0$ or $X_{buflow,i} > 0$) and $X^2_{newlow,i} > 0$ }] randomly.

drop a packet randomly from queue $W^2_{newlow,i}$. For each packet dropping increment '*lowdropped*' by 1 (i.e., *lowdropped* = *lowdropped* +1), decrement '*diff*' by 1 (i.e., *diff* = *diff* −1), decrement '*tmp*' by 1 (i.e., *tmp* = *tmp* −1).

(b) if (*tmp*=0)

Select some *i* such that [($X^1_{newhigh,i} > 1$) or ($X_{bufhigh,i} > 0$ and $X^1_{newhigh,i} > 0$)] randomly.

Drop a packet randomly from queue $W^{1}_{newhigh,i}$. For each packet dropping increment '*highdropped*' by 1 (i.e., *highdropped* = *highdropped* +1), decrement '*diff*' by 1 (i.e., *diff* = *diff* −1).

After this step $X^{2}_{newlow,i}$ becomes $X^{3}_{newlow,i}$, $W^{2}_{newlow,i}$ becomes $W^{3}_{newlow,i}$ and $X^{1}_{newhigh,i}$ becomes $X^{2}_{newhigh,i}$, $W^{1}_{newhigh,i}$ becomes $W^{2}_{newhigh,i}$.

Step 7: *selected[i]* = randomly one from $W_{bufhigh,i}$ with maximum delay,

if ($X_{bufhigh,i} > 0$)

= randomly one from $W^{2}_{newhigh,i}$, if ($X_{bufhigh,i} = 0$ and $X^{2}_{newhigh,i} > 0$)

= randomly one from $W_{buflow,i}$ with maximum delay, if ($X_{bufhigh,i} = 0$ and $X^{2}_{newhigh,i} = 0$ and $X_{bufhigh,i} > 0$)

= randomly one from $W^{3}_{newlow,i}$, if ($X_{bufhigh,i} = 0$ and $X^{2}_{newhigh,i} = 0$ and

$X_{bufhigh,i} = 0$ and $X^{3}_{newlow,i} > 0$)

= *N*+1. (i.e., none)

If selected[i] != *N*+1,

(a) if priority of the selected packet is high then

highdelay = *highdelay* + delay of *selected[i]*, *hightotaltx* = *hightotaltx* +1. Do this for each *i*. (i.e., $i = 0$ to $N-1$).

(b) else if priority of the selected packet is low then

lowdelay = *lowdelay* + delay of *selected[i]*, *lowtotaltx* = *lowtotaltx* +1. Do this step for each *i*. (i.e., $i = 0$ to $N-1$).

Step 8: Define *grandtotalsecond[i]* = $X_{bufhigh,i} + X^{2}_{newhigh,i} + X_{bufhigh,i} + X^{3}_{newlow,i}$.

```
If(grandtotalsecond[i] > b+1) define diff4=(grandtoatlsecond[i]-b+1).

while(diff4 > 0){

(a) if ( $X^3_{newlow,t}$ >0) drop packet from $W^3_{newlow,t}$. Decrement $X^3_{newlow,t}$ by 1, diff4 by 1.

(b) if( $X^3_{newlow,t}$ =0) drop a packet with minimum delay from $W_{buflow,t}$ , decrement diff4 by
    1.

}
```

After this step $X^3_{newlow,t}$ becomes $X^4_{newlow,t}$, $W^3_{newlow,t}$ becomes $W^4_{newlow,t}$

Step 9: Save remaining packets from $W^2_{newhigh,t}$,$W^4_{newlow,t}$ (after selection in the case of direct transmission) in to buffers by making sure that this buffer position is not one of '*posinuse*', for *i* =0 to *N*-1. Transmit *selected[i]*. If packets selected in this time slot are

from buffer, free corresponding buffer positions.

Step 10: Increment delay of packets within buffer by 1. Increment present timeslot by 1. If present timeslot is not equal to time slot count go to *Step* 2, else go to *Step* 11.

Step 11: Count parameters of interest

packet loss probability for high priority = *highdropped/highsum.*

packet loss probability for low priority = *lowdropped/lowsum* .

average delay for high priority = *highdelay/hightotaltx.*

average delay for low priority = *lowdelay/lowtotaltx*

Above algorithm is explained below.

In *Step* 1 *highsum*, *lowsum*, *highdropped*, *lowdropped*, *highdelay*, *lowdelay*, *hightotaltx*, *lowtotaltx* are initialized to zero.

In *Step* 2 packets after synchronization at link processor will arrive to switching fabric at the beginning of timeslot. Arrival process is uniformly distributed. It has been assumed that priority of packets coming also uniformly distributed. For each high priority packet arriving '*highsum*' is incremented by 1 and for each low priority packet arriving '*lowsum*' is incremented by 1. '*dropinside*' is initialized to 0. Controller will check in the beginning whether for any packet in the buffer the delay is greater than *b* and the packets are still required to provide delay. Controller will turn off the SOA corresponding the particular buffer position and that buffer position is not available for storage as explained in section 3.3. So controller needs to make sure that, the buffer position, which has been freed in this slot should not be used in the same slot. So controller is saving the buffer positions which have been freed in this slot in a variable '*posinuse*'. Each time controller is dropping a packet inside it is incrementing '*highdropped*' by 1 if the packet dropped is of high priority and '*lowdropped*' by 1 if the packet dropped is of low priority, '*dropinside*' by 1, and controller will note the corresponding buffer position in '*posinuse*'. This is done in *Step* 3.

It is *important* to note that *only* low priority packets may get dropped inside the loop in this algorithm. This will happen because if a low priority packet, which is in buffer and have already recirculated *b* number of recirculations. But in the present time slot a high priority packet have arrived for the same destination then the newly arrived high priority packet will be transmitted in this particular time slot, so the low priority packet from buffer has to remain in buffer in this slot also. So in the next time slot controller will find this low priority packet has rotated more than *b* recirculations, and will drop the packet by turning off the SOA corresponding to the buffer position. High priority packet will be *never* dropped inside.

In *Step* 4, controller will form logical queues as explained in figure 5.1. Controller will now check whether sum of (a) high priority packets from buffer to i (b) newly arrived high priority packets to i (c) newly arrived low priority packets to i, represented by '$total_i$' is greater than b+1. If '$total_i$' is greater than b+1, controller will be dropping packets from $W_{newlow,i}$ till limit is achieved, if there are no packets left in $W_{newlow,i}$ and limit is still not achieved then controller will drop packets from $W_{newhigh,i}$ till limit is reached. For each dropping low priority packet from $W_{newlow,i}$ controller will increment '*lowdropped*' by 1 and for each dropped high priority packets controller will be incrementing '*highdropped*' by 1. After this step $X_{newlow,i}$ becomes $X^{1}_{newlow,i}$ ($0 \leq X^{1}_{newlow,i}$

$\leq X_{newlow,i}$)and $W_{newlow,i}$ becomes $W^1_{newlow,i}$(may be same as $W_{newlow,i}$) and $X_{newhigh,i}$ becomes $X^1_{newhigh,i}$ ($0 \leq X^1_{newhigh,i} \leq X_{newhigh,i}$) and $W_{newhigh,i}$ becomes $W^1_{newhigh,i}$ (may be same as $W_{newhigh,i}$) . Do Step 4 for each *i* (i.e., *i*=0 to *N*-1).

In Step 5, controller will check whether sum of all the logical queues (recently) for a particular '*i*', is exceeding *b*+1. If sum of these logical queues for any particular '*i*' is exceeding *b*+1 controller will try to achieve this limit by dropping newly arrived low priority packets till limit is reached or there are no more newly arrived low priority packets for that destination.

After this process $X^1_{newlow,i}$ becomes $X^2_{newlow,i}$ ($0 \leq X^2_{newlow,i} \leq X^1_{newlow,i}$), $W^1_{newlow,i}$ becomes $W^2_{newlow,i}$ (may be same as $W^1_{newlow,i}$).

In *Step* 6, first number of buffers required in different cases for any '*i*' th output is evaluated, then total number of buffers required is calculated as sum of buffers required for all *i*'s. Controller will increment '*selectedlow*' by 1 for each packet that will be going to be transmitted from $X^2_{newlow,i}$. '*selectedlow*' is required in case of dropping newly arrived low priority packets when sufficient number of free buffers not available to save contending packets for various output lines as explained below. Controller will check the number of free buffers available, by making sure that the buffers which are freed in this slot (by dropping packets inside as explained in *Step* 3) are not be treated as free buffers available for storage (that is why there is '*dropinside*' term is present in '*freebuf*' equation). Controller will then check whether free buffers available are sufficient to save the packets contending for different outputs, if they are not sufficient controller will define a variable '*tmp*' defined as '$\sum_{i=0}^{N-1} X^2_{newlow,i}$ - *selectedlow*'. Controller will start dropping newly arrived low priority packets by randomly choosing some '*i*', by making sure that there exist at least one packet (if any is there)for each '*i*'. Each time a newly arrived low priority packet is dropped '*tmp*' decrements by 1. If '*tmp*' has reached 0 and still free buffers are not there to save newly arrived high priority contending packets, newly arrived high priority packets are randomly dropped from $W^1_{newhigh,i}$ by making sure that there exists at least (if any there) for transmission on '*i*'. After this step $X^2_{newlow,i}$

becomes $X^{3}_{newlow,i}$ $(0 \leq X^{3}_{newlow,i} \leq X^{2}_{newlow,i})$, $W^{2}_{newlow,i}$ becomes $W^{3}_{newlow,i}$ (may be same as $W^{2}_{newlow,i}$) and $X^{1}_{newhigh,i}$ becomes $X^{2}_{newhigh,i}$ $(0 \leq X^{2}_{newhigh\,i} \leq, X^{1}_{newhigh\,i})$, $W^{1}_{newhigh,i}$ becomes $W^{2}_{newhigh,i}$ (may be same as $W^{1}_{newhigh,i}$).

In *Step* 7, packets that to be transmitted are selected according to the priority, maximum delay conditions randomly. Delays of selected packets depending on their priorities are added to '*lowdelay*' (if low priority packet is selected) or '*highdelay*' (if high priority packet is selected) depending on the case. Similarly '*hightotaltx*' or '*lowtotaltx*' are incremented by 1 depending on the priority of packet selected for transmission.

In *Step* 8, controller will check whether sum of all packets to a particular destination exceeding b+1, if this is true, controller will drop packets from $W^{3}_{newlow,i}$ till limit is achieved or there are no more packets in $W^{3}_{newlow\,i}$. If there are no more packets in $W^{3}_{newlow,i}$ controller will drop packets from $W_{buflow,i}$ till limit is achieved. After this step $X^{3}_{newlow,i}$ becomes $X^{4}_{newlow,i}$ $(0 \leq X^{4}_{newlow,i} \leq X^{3}_{newlow,i})$, $W^{3}_{newlow,i}$ becomes $W^{4}_{newlow,i}$ (may be same as $W^{3}_{newlow,i}$).

In *Step*9, first high priority packets that are there in logical queue $W^{2}_{newhigh\,i}$ are saved in to the buffers by making sure that the buffer position is not one in '*posinuse*'. Save remaining packets from $W^{4}_{newlow,i}$ in to buffers by making sure that this buffer position is not one of 'posinuse', for i = 0 to N-1. Transmit selected packets in *Step* 7. If packets selected in this time slot are from buffer, free corresponding buffer positions.

Controller will increment delay for the packets saved in buffers before going for processing in next time slot. This is repeated. *Packet loss probability*, *average delay* for both the priority classes are calculated as explained in *Step*s 10, 11 of the algorithm.

Chapter 6

Results and Discussion

In this Chapter simulation results of the work carried out are presented. First a comparison of two switching strategies for the all-optical switch based on multiwavelength fiber loop memory i.e., (1) 'drop at inputs' switching strategy and (2) 'drop inside' switching strategy is given. Comparison is made by varying 'recirculation limit' (b), total number of buffers (B), load (p). Simulation is carried out by assuming high arrival rates as these switches, which will generally be encountered in backbone networks. It is also assumed that due to noise limitations it is not possible to employ large number of buffers (B) in the switch. Then the performance of the proposed priority algorithm for the switch and its effectiveness in minimizing packet loss probability for high priority packets (which are loss sensitive) relative to low priority packets (which are loss insensitive) is evaluated, by comparing packet loss probability and delay for both the arrival classes with the case when no priority is there in the switch.

6.1Comparison of 'drop inside' and 'drop at the inputs' strategies

(i) *Varying 'recirculation limit'* (b)

For a given buffer size (B) and load (p) in both the strategies when 'recirculation limit' (b) is very small, packet loss probability is very high and average delay is very low. This is because, even if there are sufficient number of buffers (may be $B \geq Nb$ when b is small) are available, the switch can not store packets for any particular output more than 'recirculation limit' (b), and hence more number of packets will be dropped. As queue size for any output will not exceed b and b is small, average delay remains small when b is small in both the strategies. At these values of b (i.e., small values of b) both the strategies are performing similarly. This can be observed from figures 6.1 to 6.6. In these figures packet loss probabilities are plotted on logarithmic scale. As b increases, packet loss probability decreases till a particular b value is reached (this value depends on B and p) after which packet loss probability remains constant in both the switching strategies.

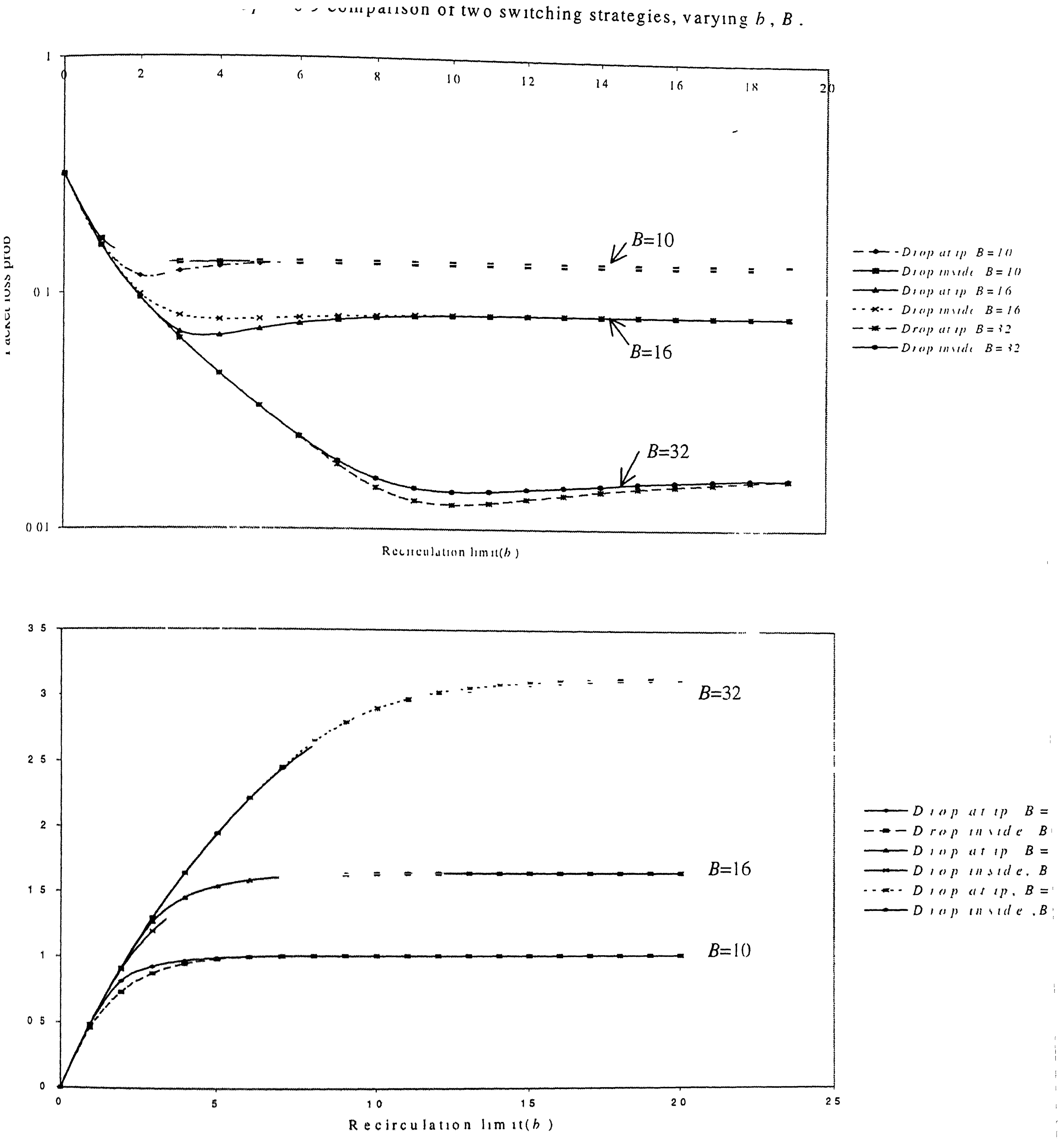
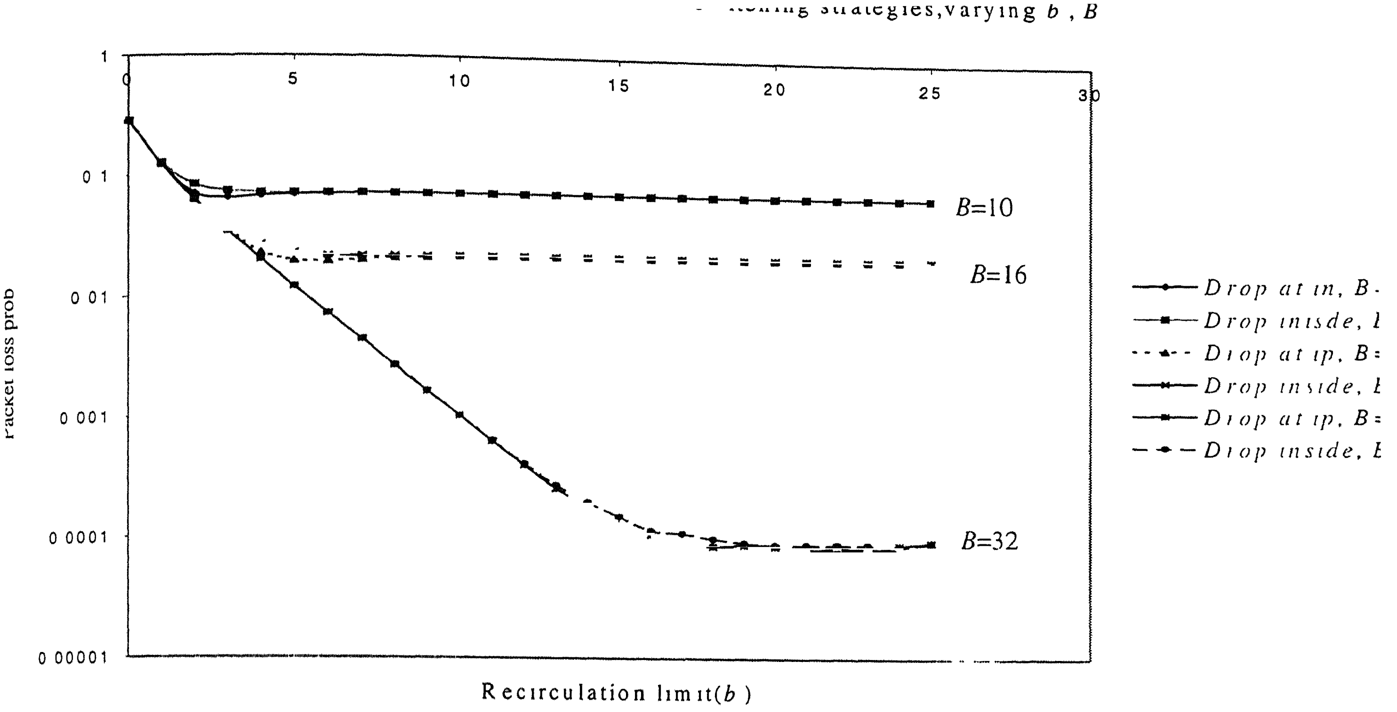


Fig 6.1: Comparison of 'drop inside' and 'drop at inputs' switching strategies, at p=0.9, varying b, B.



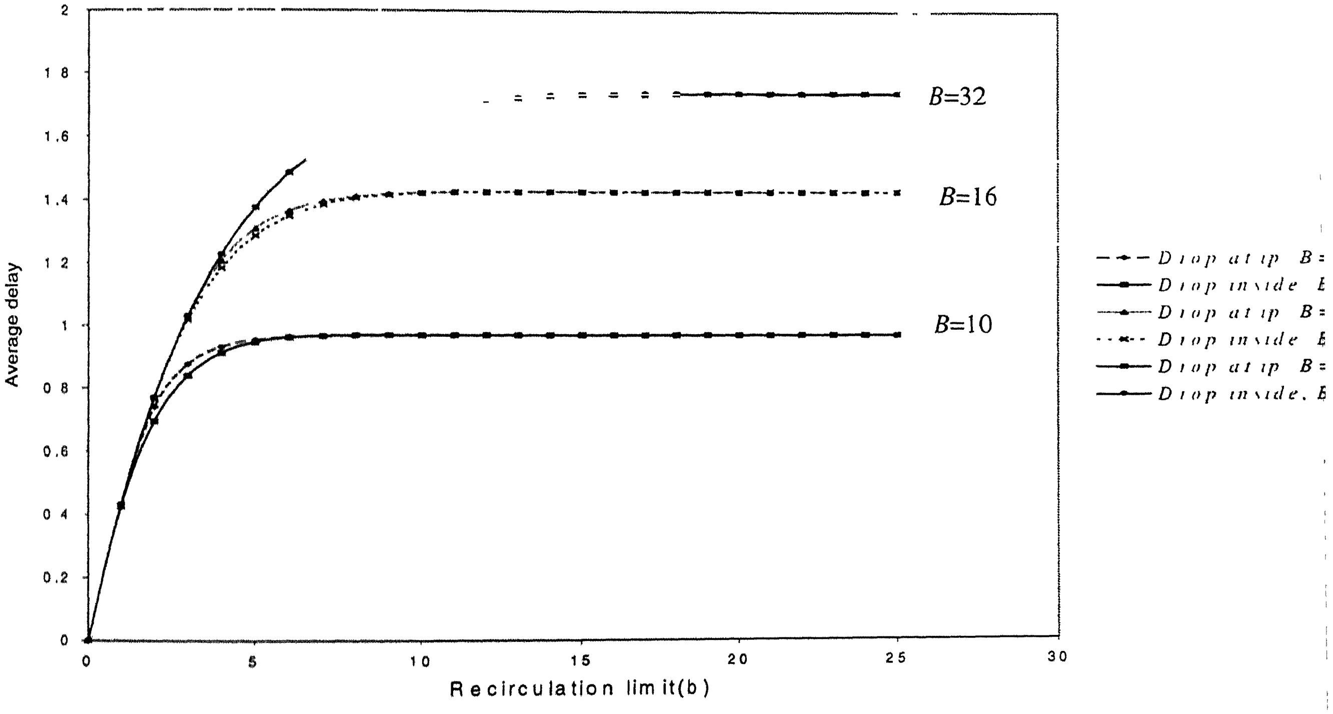


Fig 6.2 Comparison of 'drop inside' and 'drop at inputs' switching strategies, at p=0.8, varying b,B.

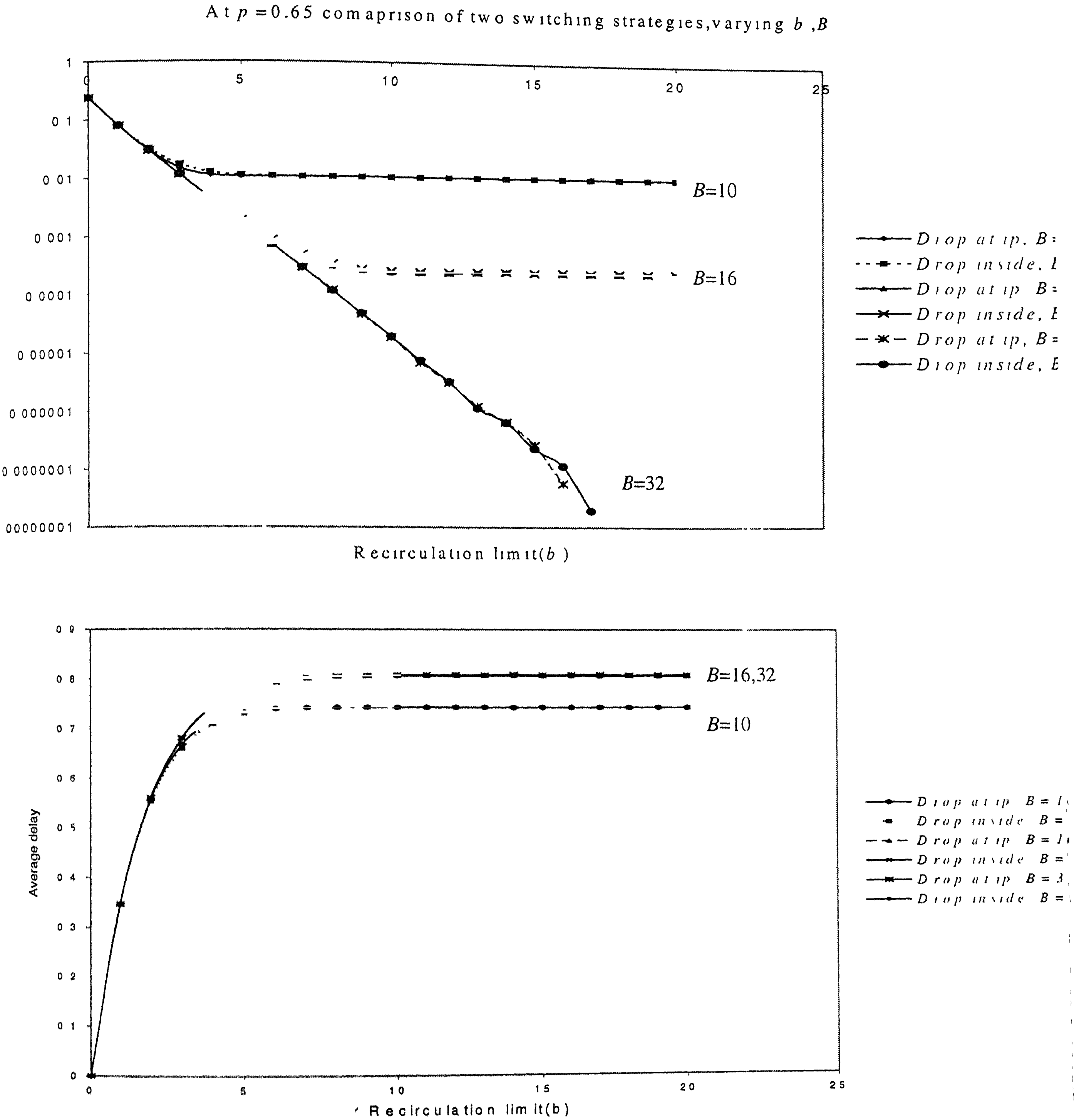


Fig 6.3: Comparison of ‘drop inside’ and ‘drop at input’ switching strategies, at p=0.65, varying b, B.

For a given set of p, B the switch can not provide packet loss probability less than this particular packet loss probability irrelevant of recirculation limit (b). Coming to average delay, in both the strategies it increases and reaches a constant value, as now packets that can be stored for any particular output are large. This implies that for a given set of p, B packets for any output port will never experience delay more than this value. The value of b, at which this constant packet loss probability and average delay occurs, depends on B and p. For this range of values of b there exists a difference in packet loss probabilities, average delays of both the switching strategies. These differences actually depend on p and B. Similar results are obtained for other values of p. Figure 6.2 shows average delay and packet loss probability variation with b for p=0.8. In figure 6.3 after some value of b, packet loss probability becomes very low. The simulation time needed to estimate the probability is observed to become very large. Hence it is difficult to estimate the probability. One can only say it is very low. This corresponds to infinity in logarithmic scale hence no value will be shown. Due to large and impractical computational time required, confidence intervals are not calculated for these results.

In general packet loss probabilities that can be achieved by 'drop at inputs' switching strategy are less than those of 'drop inside' strategy. For the range of b values which this happen average delays are more in 'drop at inputs' switching strategy than in 'drop inside' switching strategy. These differences in packet loss probability, average delays are very small. Except these differences both the switching strategies perform similarly.

(ii) *Varying 'buffer size'* (B)

As buffer size B increases packet loss probability in both the switching strategies decreases and average delay in both of them increases. For a given p, as 'B' increases the value of b, at which packet loss probability, average delay becomes constant, as mentioned above will be increasing (figures 6.1 to 6.3). As B increases packet loss probability difference of both switching strategies (for a given p and b where the difference exists) remains almost same in both switching strategies, but the difference in delay decreases. This can be observed in figures 6.1 to 6.3.

(iii) *Varying 'load'* (p)

As load increases packet loss probability, average delay for a packet increases in both the switching strategies. For a given B, as load (the arrival rate) increases the value of b, at

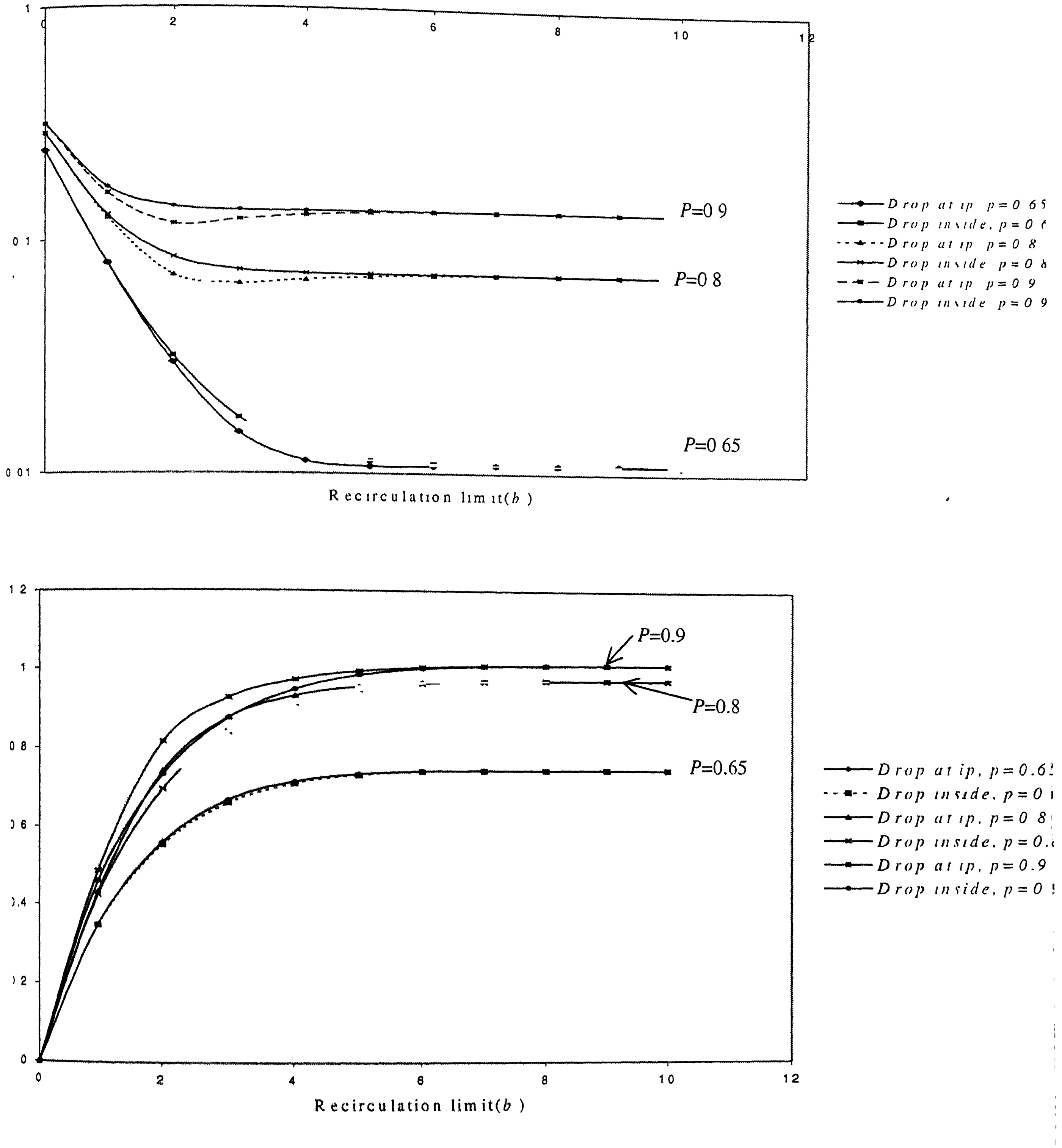


Figure 6.4: Comparison of 'drop inside' and 'drop at inputs' switching strategies at B=10, varying b, p.

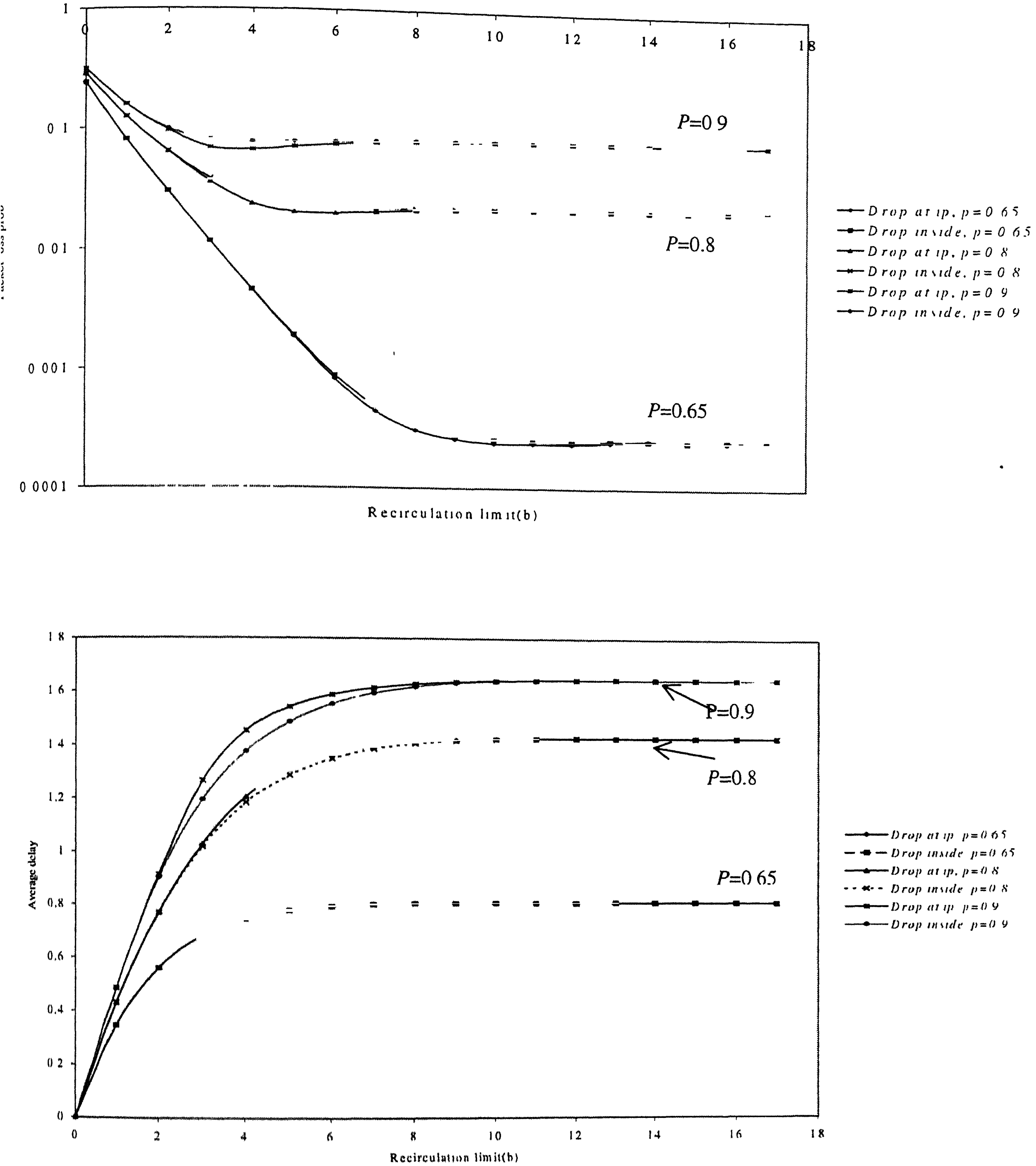


Fig 6.5: Comparison of 'drop inside' and 'drop at inputs' switching strategies, at B=16, varying b, p.

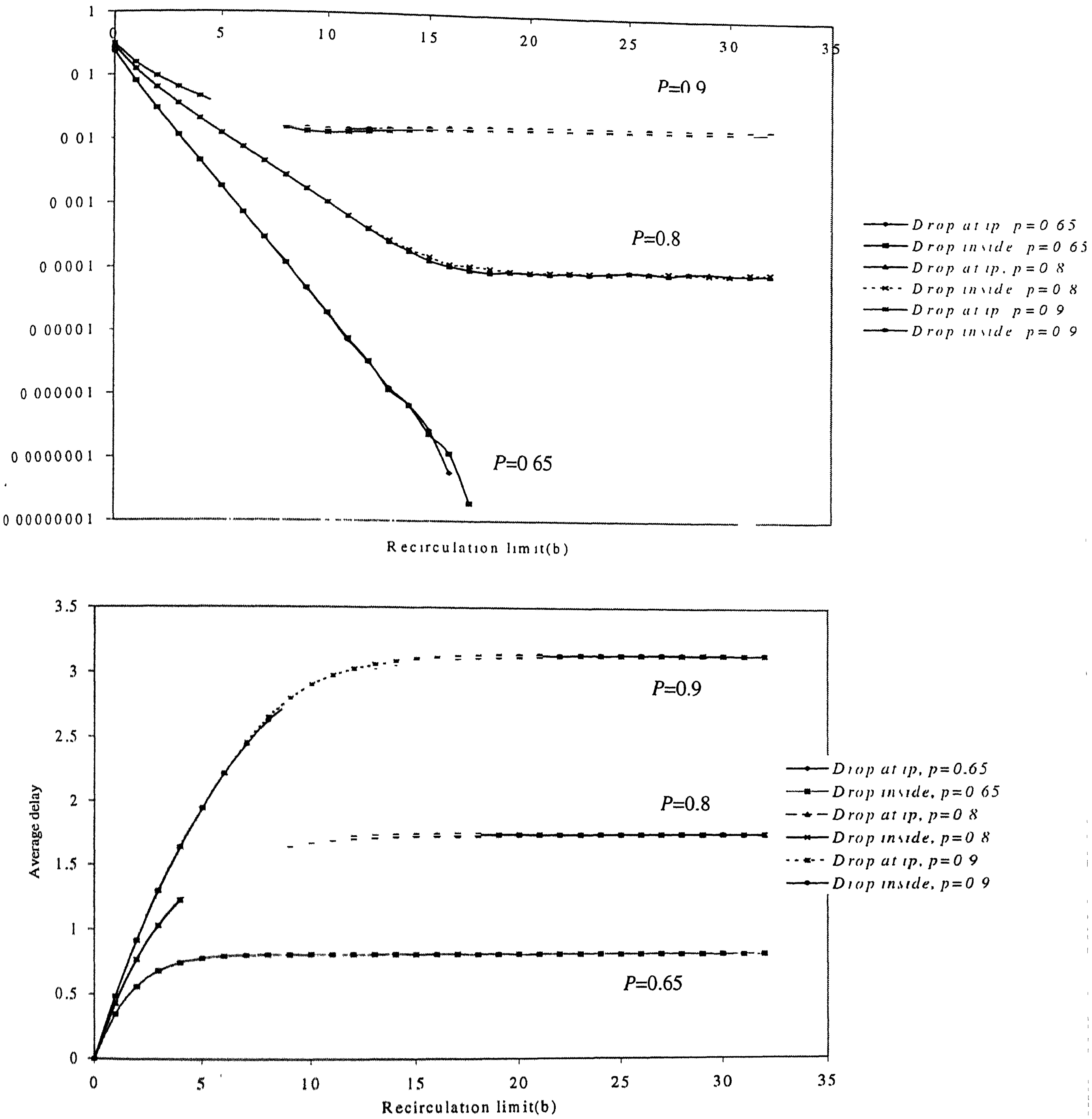


Fig 6.6: Comparison of 'drop inside' and 'drop at inputs' switching strategies at B=32, varying b, p.

which packet loss probability, average delay becomes constant, as mentioned above will be increasing. This can be observed from figures 6.4 to 6.6. Further as load increases the difference of packet loss probabilities in both switching strategies (for a given *B* and *b* where the difference exists) remains almost same, but the difference in delay increases slightly.

From simulation results shown in figures 6.1 to 6.6, it can be said that both the switching strategies are performing almost similarly except with some minor differences.

6.2 Performance of priority switching algorithm

In this section the performance of the proposed priority switching algorithm in reducing packet loss probabilities for the loss sensitive high priority packets relative to loss insensitive low priority packets is presented, by comparing it with 'without priority' case in which both the classes (high priority, low priority) are given same priority. Effects of varying *b*, *B* and *p* on priority algorithm are presented.

(i) *Varying 'recirculation limit'* (*b*)

High priority packets will suffer low packet loss probabilities relative to low priority packets for all range of *b* values. When recircualtion limit (*b*) is very small packet loss probabilities for both high priority and low priority classes using priority algorithm (from now this will be called Case 2) are large. But when these are compared with 'without priority' case (from now this will be called Case 1), packet loss probability for high priority packets is smaller, and that of low priority packets is larger. Coming to average delay for a packet, when *b*=0 delay is zero for both the classes of arrivals (i.e., high priority as well as low priority) in both the Cases (i.e., priority and without priority). As *b* increases packet loss probability for both the classes of arrivals goes down and reaches a constant value in Case 2 as in Case 1.

Packet loss probability for high priority packets attains a minimum value as *b* is increasing from lower values, and remains at this minimum value for a range of values of *b*, and then increases with *b* to reach the constant value. This happens because initially packet loss probability is higher as, *b* values are very small and more packets are dropped. As *b* increases packet loss probability goes down and reaches a minimum value, as high priority packets will find sufficient number of free buffers on arrival. The range of values of b for which packet loss probability for high priority packets remains at minimum value depends on p, B. As b increases further packet loss probability for high priority packets starts increasing and reaches the constant value (which depends on p, B). This is because high priority packets on arrival will not find sufficient number of free buffers. The constant value at which high priority packet loss probability remains is still very small compared with Case 1. But the price paid for this improvement is that there is an increase in packet loss probability values for low priority packets compared to Case 1. This is because high priority packets are always given priorities in case of transmission over low priority packets, and high priority packets if finds position in buffer and total number of packets for any destination exceeds b low priority packets will be dropped from buffer. This can be seen in figures 6.7 to 6.17. In these figures to show the effectiveness of proposed algorithm in minimizing packet loss probability for high priority packets relative to low priority packets as compared to Case 1, it has been assumed that both the arrival classes (i.e., high priority, low priority) are coming with equal probabilities (In practical conditions arrival rates of high priority packets are smaller than low priority packet arrival rates). Packet loss probabilities are plotted on logarithmic scale and as packet loss probabilities of high priority packets in figures 6.10, 6.12, 6.17are reaching to very low values for some range of values of b, there is no packet loss probability in this range.

Coming to average delay for a packet, high priority packets in 'priority' algorithm (Case 2) will experience much smaller delays than in 'without priority' algorithm (Case 1). But average delay for a packet of low priority packets in 'priority' algorithm goes higher than in 'without priority' algorithm.

From the results shown in figures 6.7 to 6.17 one can say that proposed priority switching algorithm for the switch is effective in minimizing packet loss probability, average delay of high priority packets relative to low priority packets than in 'without priority' Case.

(ii) *Varying 'buffer size'* (B)

At a given p, as buffer size (B) increases packet loss probabilities go down for both the arrival classes as in Case 1. The minimum packet loss probability value for high priority packets using 'priority' algorithm as discussed above go down with increasing B values.

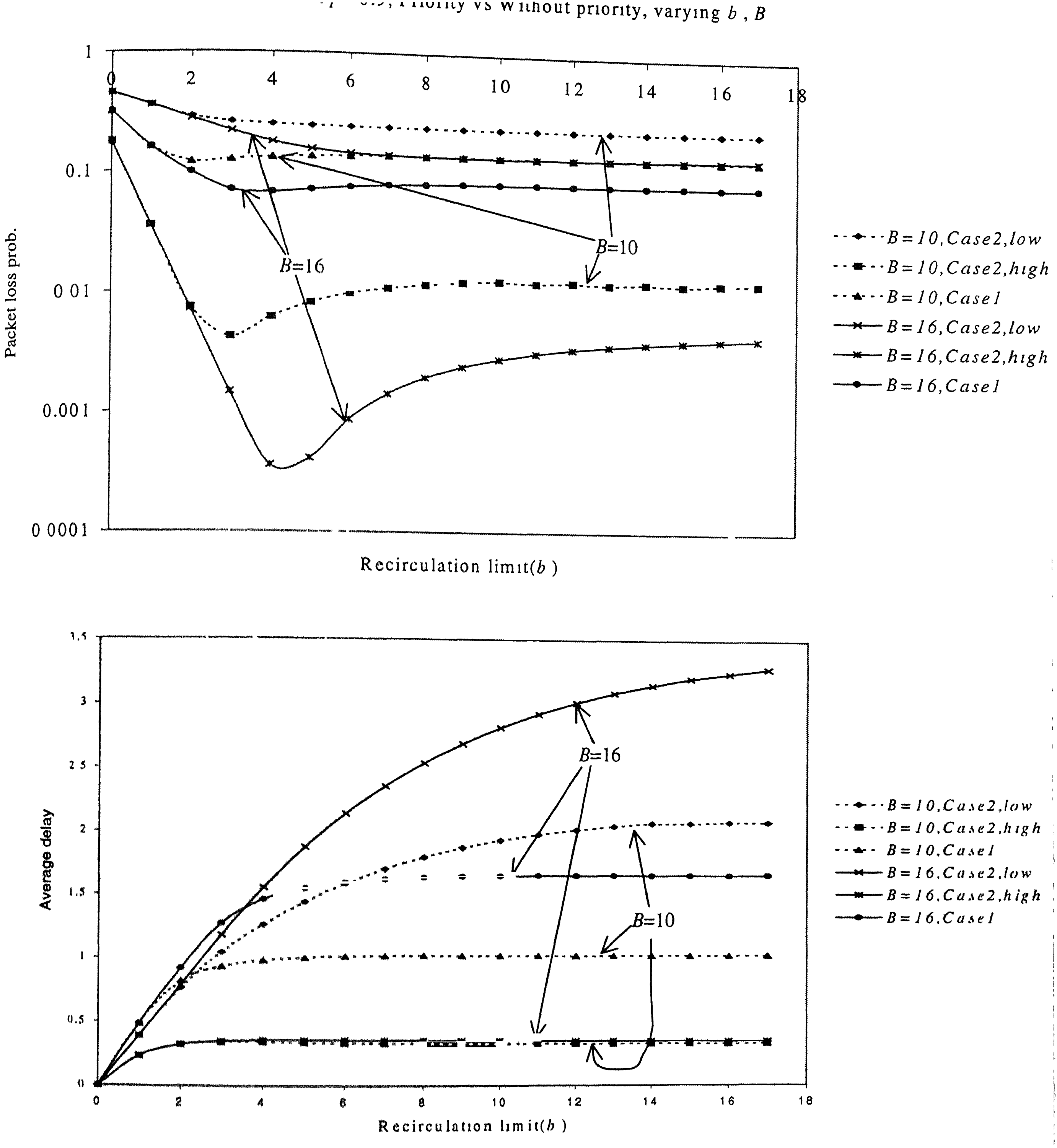


Fig 6.7: Priority vs Without priority at $p = 0.9$, varying b, B.

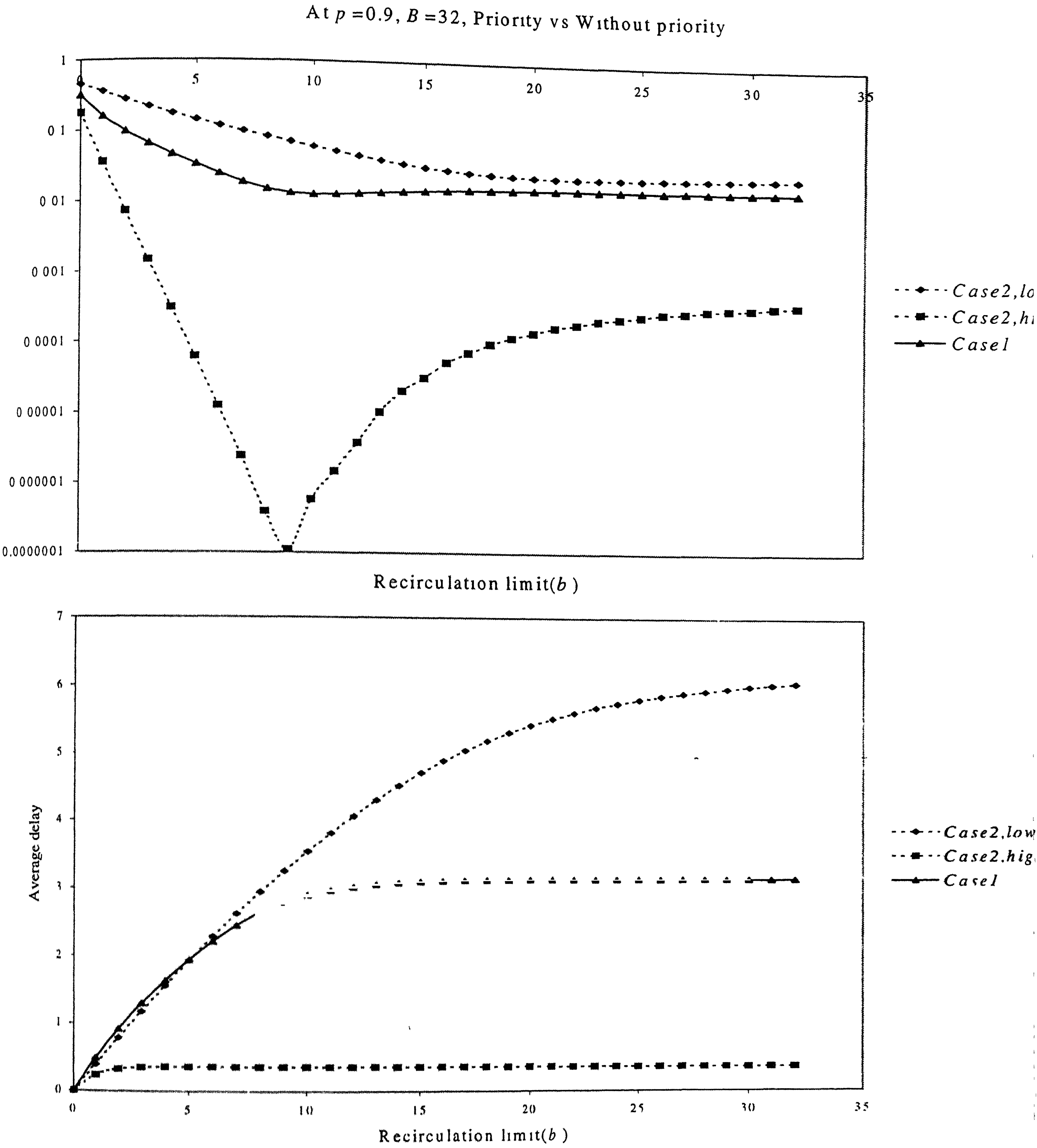


Fig 6.8: Priority vs Without priority at p =0.9, B =32, varying b.

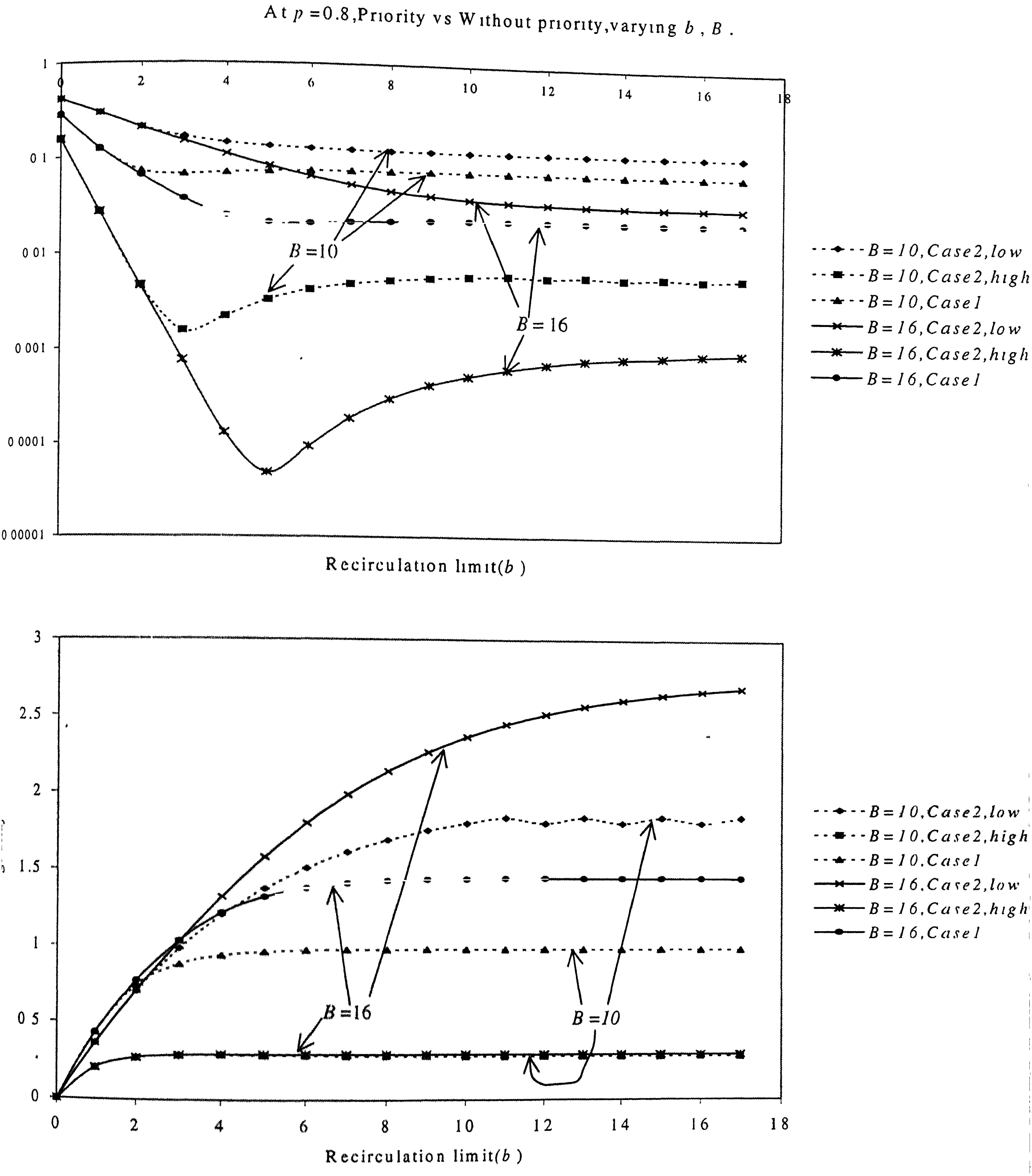


Fig 6.9: Priority vs Without priority at $p = 0.8$, varying b, B.

priority,varying b .

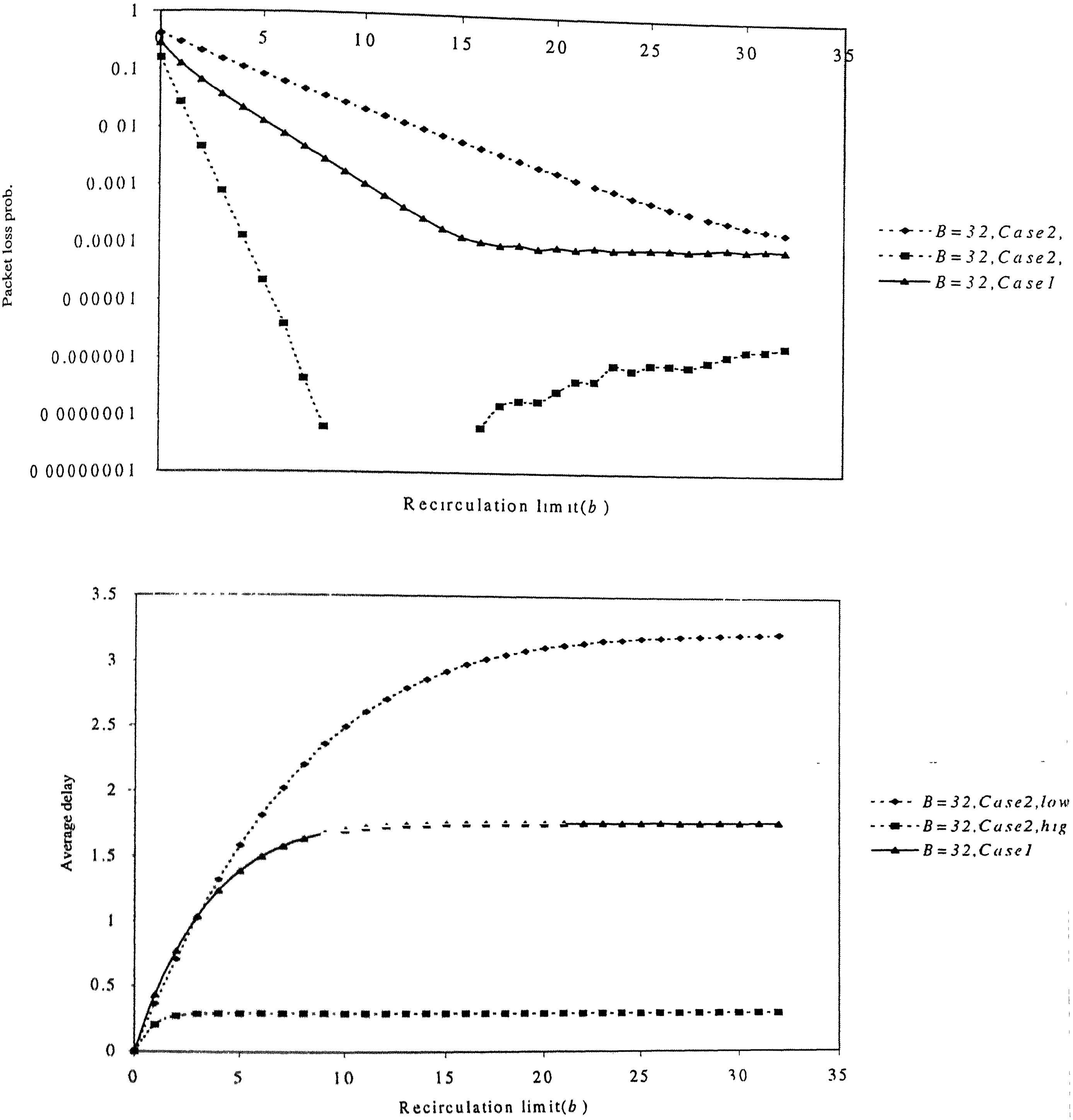


Fig 6.10: Priority vs Without priority at p=0.8, B=32, varying b.

At p =0.65,Priority vs Without priority, varying b ,B .

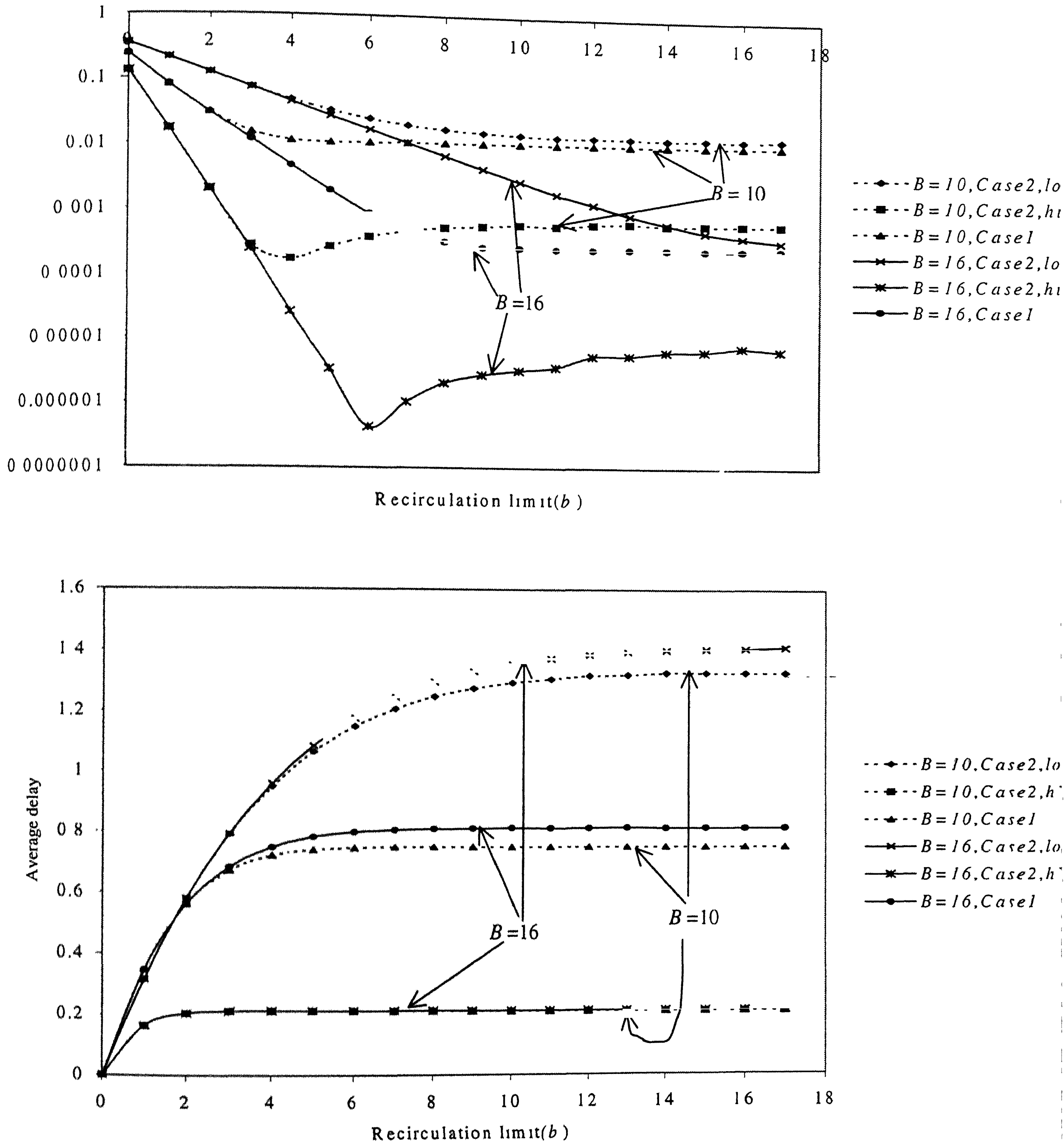


Fig 6.11: Priority vs Without priority at $p = 0.65$, varying b, B.

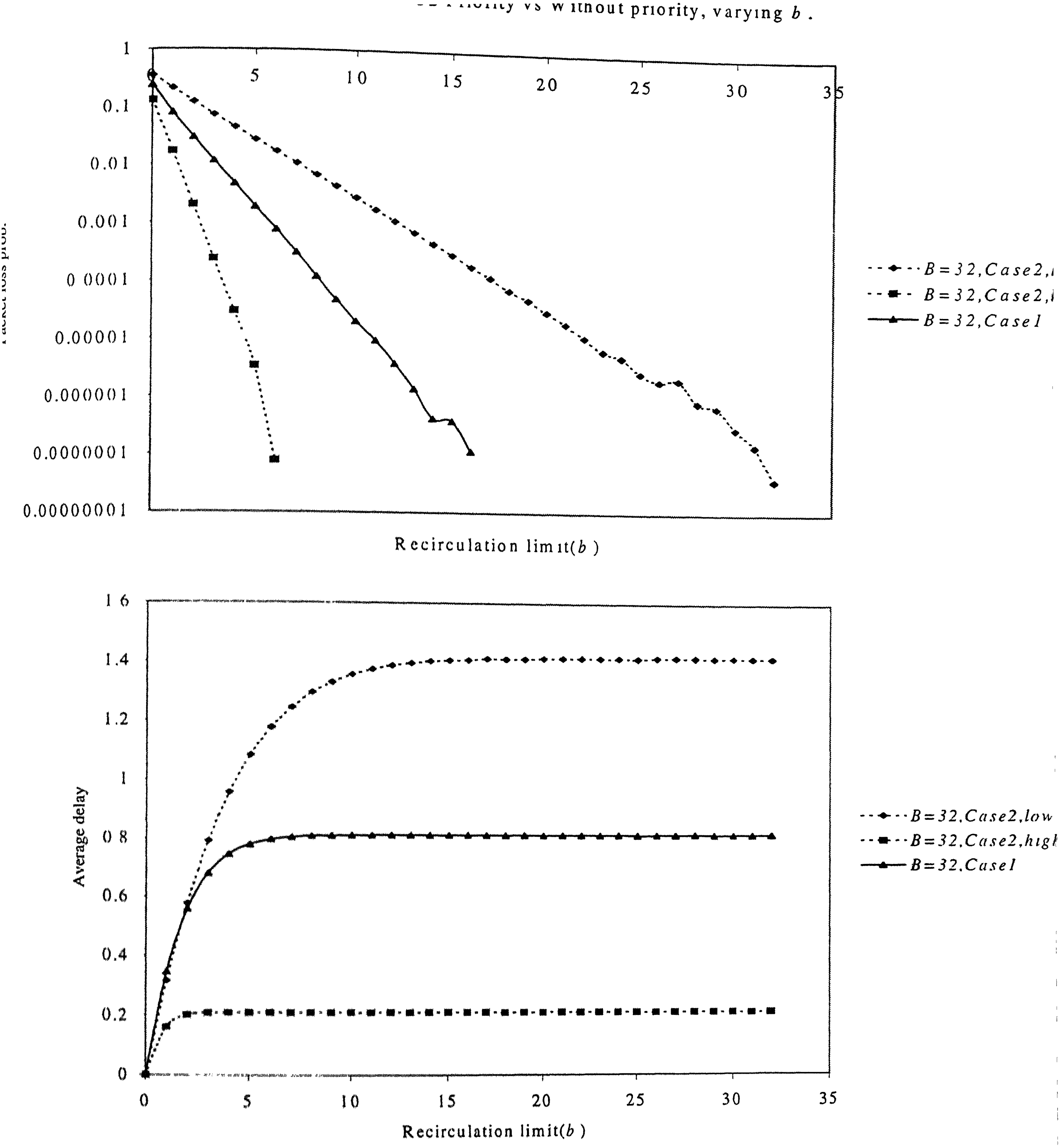


Fig 6.12: Priority vs Without priority at p=0.65, B=32, varying b.

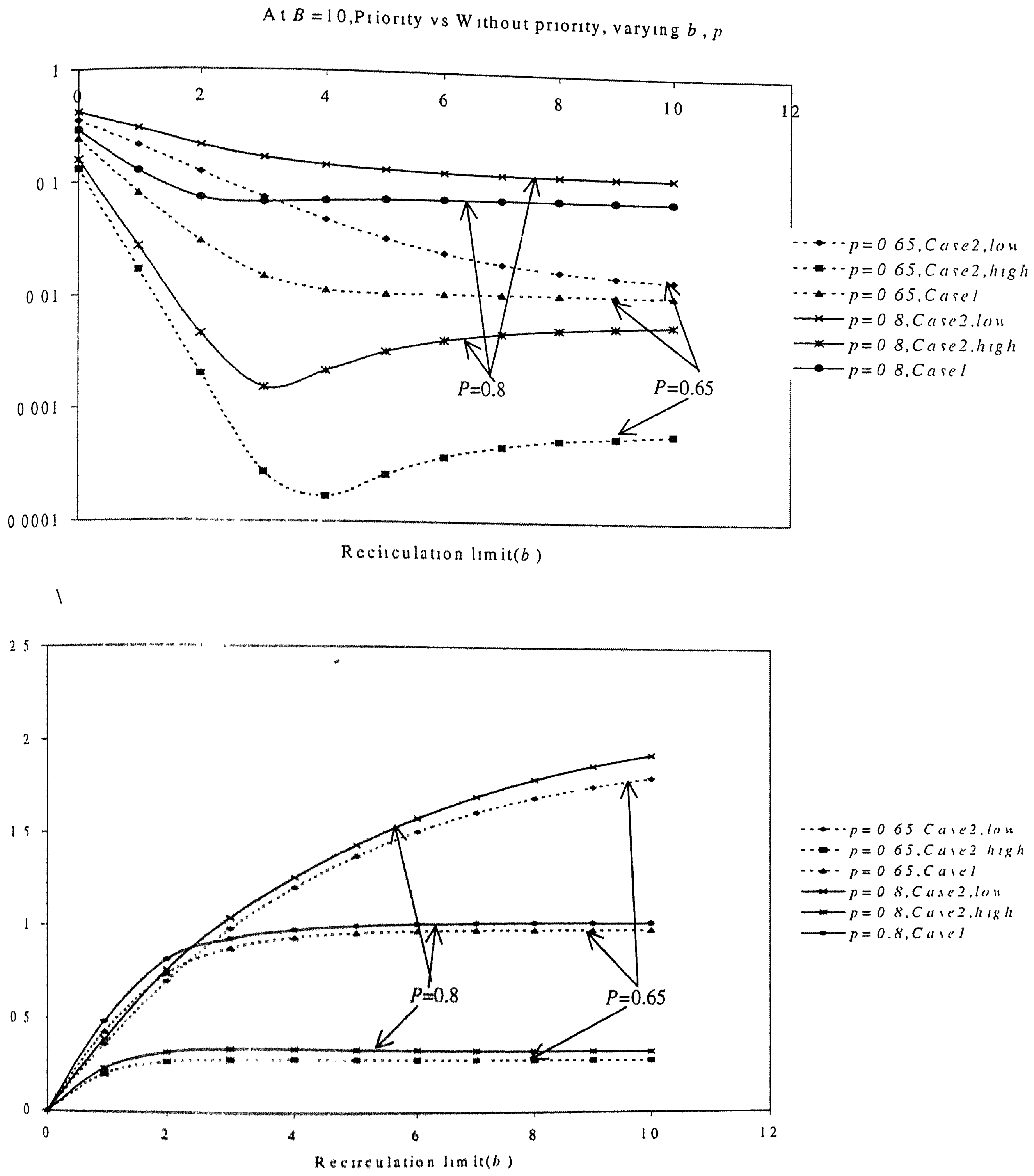


Fig 6.13: Priority vs Without priority, at B=10, varying b, p.

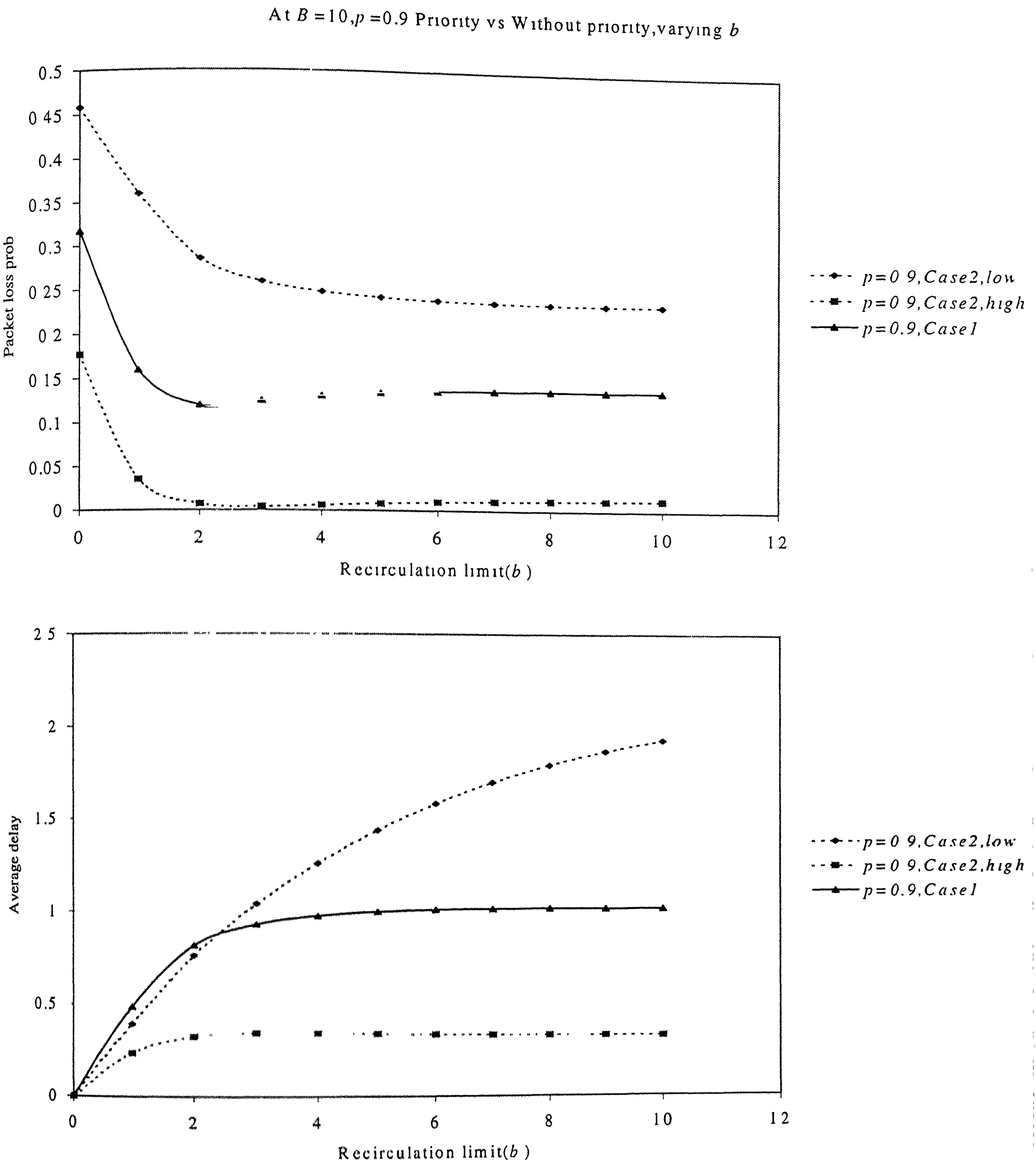


Fig 6.14: Priority vs Without priority B=10, p=0.9, varyıng b.

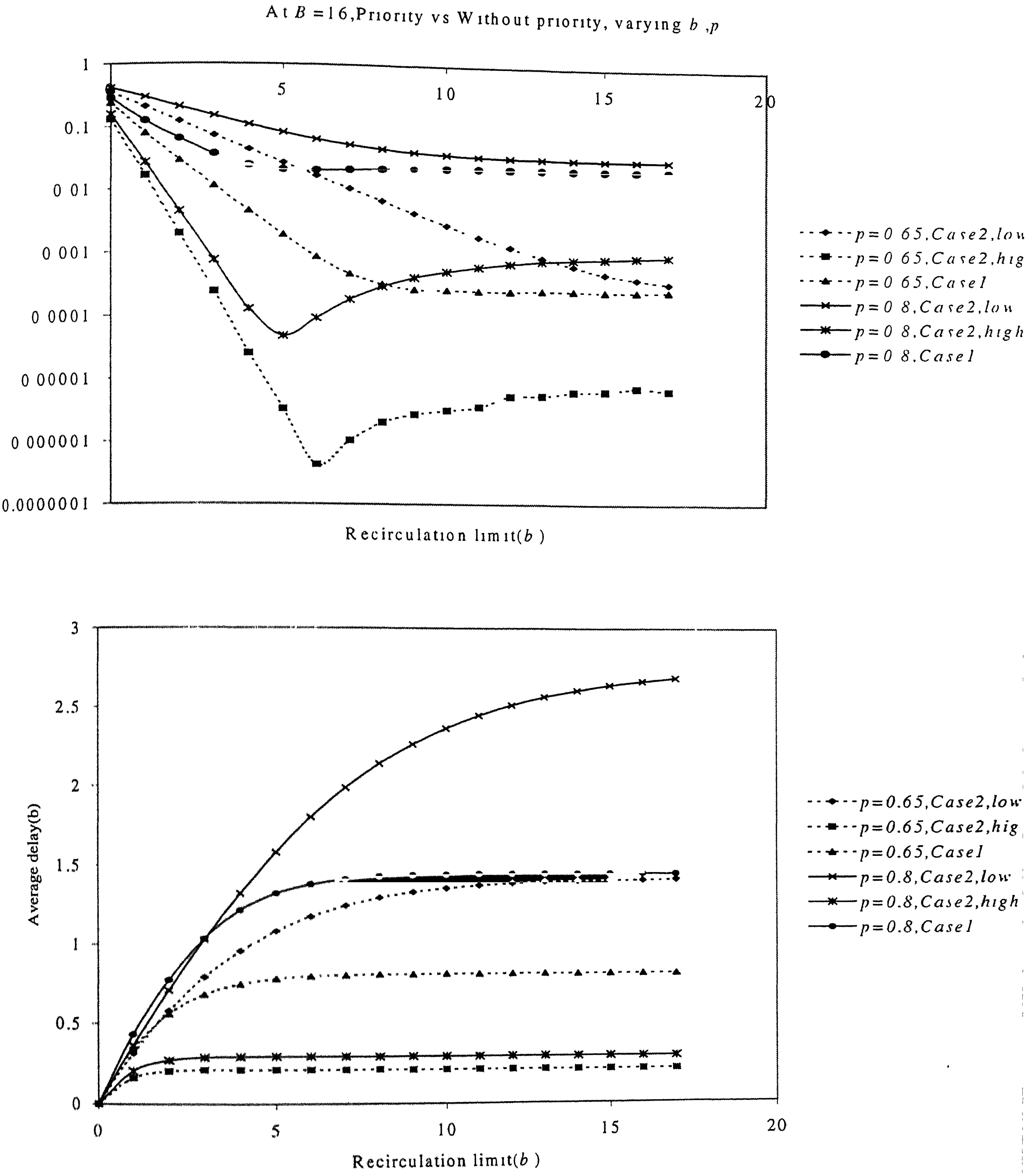


Fig 6.15: Priority vs Without priority, at B=16, varying b, p.

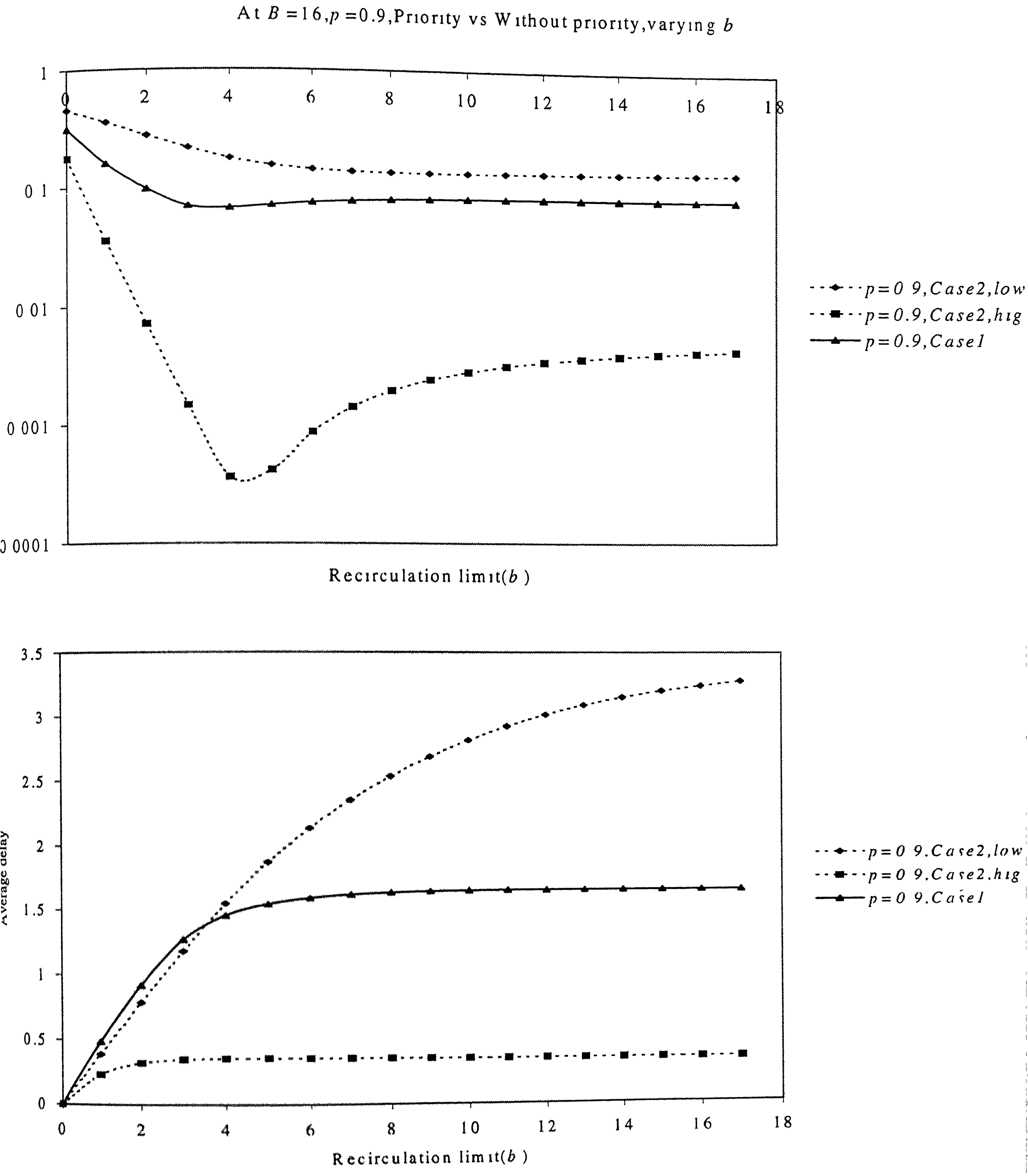
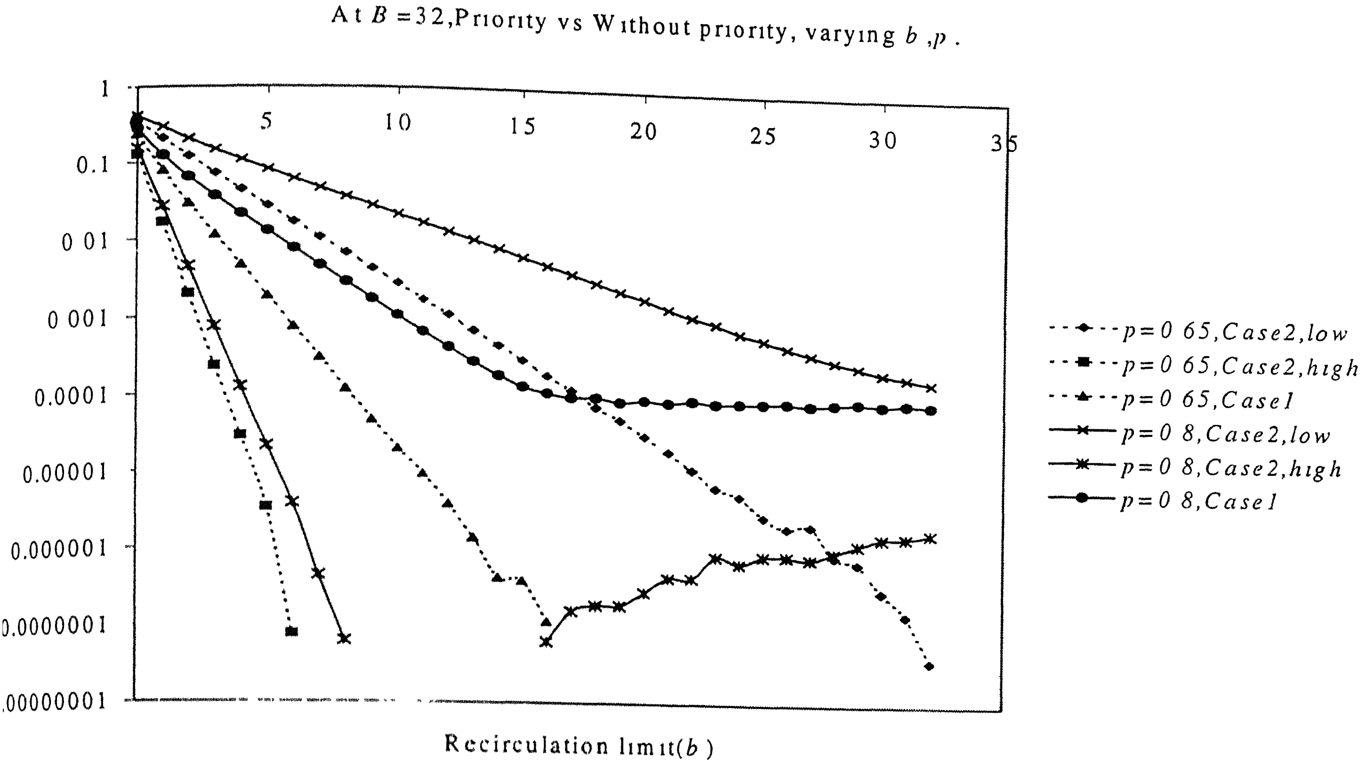


Fig 6.16:Priority vs Without priority, p=0.9,B=16,varying b.



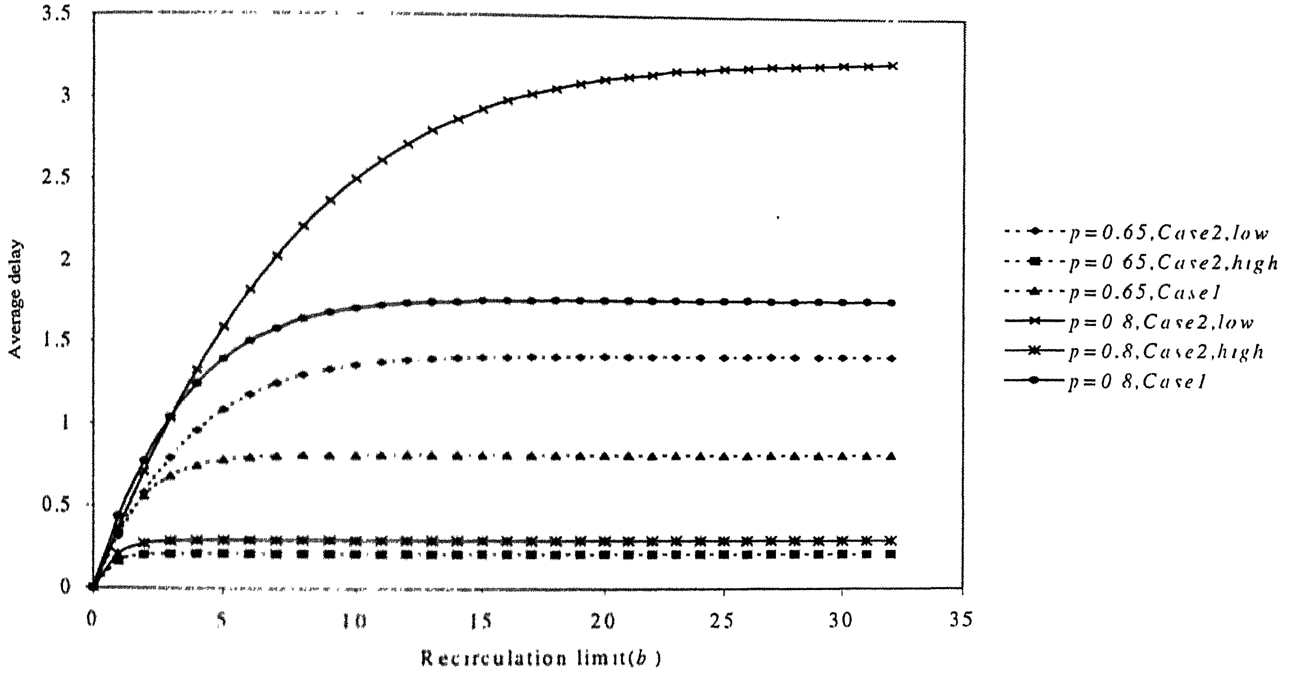


Fig 6.17: Priority vs Without priority, at B=32, varying b, p.

Coming to average delay for a packet, with increasing B, increase in delay for high priority packets will be very small. But the increase in average delay for low priority packets will be larger with increasing B. This can be observed from figures 6.7 to 6.12.

(iii) *Varying 'load'* (p)

At a given B, as load (p) increases packet loss probabilities and average delays go up for both the arrival classes as in 'without priority' Case. The minimum packet loss probability value for high priority packets using 'priority' algorithm as discussed above go up with increasing p values. With increasing p, increase in delay for high priority packets will be very small, but the increase in average delay for low priority packets will be larger with increasing p. This can be observed from figures 6.13 to 6.17.

Chapter 7

Conclusion and Future scope

This work proposes switching strategies for the all-optical switch based on multiwavelength fiber loop memory, which uses shared buffer with and without priority traffic.

We will first conclude that for a given buffer size (*B*), and load (*p*) it is not possible to achieve packet loss probability less than certain minimum value and there is no need to have recirculations more than certain minimum value. Our second observation is that both the switching strategies considered, i.e., 'drop inside', 'drop at inputs' switching strategies perform almost similarly. We will finally conclude that, it is not possible to employ 'push out' priority scheme in the switch. Proposed priority switching algorithm is effective in minimizing packet loss probability of high priority packets relative to low priority packets, than in 'without priority' case.

Future scope

Present work can be extended in many ways. Some possible directions are:

- Issues regarding synchronization can be focused.
- Possibility of implementing multicasting with switch can be investigated.
- Mathematical modeling of the switching strategies considered can be investigated.
- Noise analysis of the switch can be done to determine possible number of recirculations for a given buffer size.

References

1. Eric Nussbaum , " Communication Network Needs and Technologies – A place for photonic switching ? ," *IEEE Journal on Selected Areas in Communication,* August 1988, pp 1036-1043.

2. F Masetti *et al.,* "High speed, high capacity ATM optical switches for future telecommunication transport networks," *IEEE Journal on Selected Areas in Communication,* June 1996, pp. 979-996.

3. Piero Gambini *et al.,* "Transparent optical packet switching: Network architecture and Demonstrators in the KEOPS project," *IEEE Journal on Selected Areas in Communication,* September 1998, pp. 1245-1258.

4. Hamid Ahmadi and Wolfgang E. Denzel, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communication,* September 1989, pp. 1091-1103.

5. Paul E. Green, Jr., *Fiber Optic Networks*, Prentice-Hall, Englewood cliffes, NJ, 1993.

6. David K. Hunter et al, "Buffering in optical packet switches," *IEEE Journal of Lightwave Technology,* December 1998, pp. 2081-2094.

7 Daniel J. Blumenthal et al, Editors, *IEEE Journal of Lightwave Technology,* December 1998.

8. Martin de Prycker, *Asynchronous Transfer Mode Solution for Broadband ISDN*, Prentice-Hall, 1995.

9. M. J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input vs. output queueing on a space-division packet switch," *IEEE Trans. Communications, vol. COM-35,* December 1987, pp. 1347-1356.

10. Daniel J. Blumenthal et al, "Photonic packet switches: Architectures and experimental implementations," *Proceedings of the IEEE,* November 1994, pp. 1650-1667.

11. Rajiv Ramaswami and Kumar N. Sivarajan, *Optical networks: A practical perspective*, Morgan Kaufmann publishers, Inc. San Francisco, California, 1998.

12. Hussein T. Moftah, Jaafar M.H. Elmirghani, *Photonic switching technology systems and networks*, (A selected reprint volume), IEEE press, 1998.

13. Mohammed Atiquzzaman, "Buffer dimensioning in ATM networks using no-priority and pushout space priorities," *proceedings of GLOBECOM 1995, IEEE,* 1995, pp. 388-392.

14. R.Y. Awdeh and H.T. Mouftah, "Survey of ATM switch Architectures," *Communication Networks and ISDN systems, Vol. 27,* November 1995, pp. 1567-1613.

15. Abhijit K. Choudhury and Ellen L. Hahne, "Space priority management in a shared memory ATM switch," *proceedings of GLOBECOM 1993, IEEE,* 1993, pp. 1375-1383.

16. Pankaj Dwivedi, *Performance analysis of all-optical packet switch with header replacement mechanism*, M.Tech.Thesis, Dept. of Electrical Engg., IIT, Kanpur, July, 1999.

17. Michel G. Hluchyj and Mark J. Karol, "Queueing in High-Performance Packet switching," *IEEE Journal on Selected Areas in Communication,* September 1988, pp 1587-1597.